

libstdc++

Generated by Doxygen 1.7.1

Mon Jul 19 2010 22:44:11

Contents

1	Todo List	1
2	Module Documentation	11
2.1	Extensions	11
2.1.1	Detailed Description	12
2.2	SGI	12
2.2.1	Detailed Description	15
2.2.2	Function Documentation	16
2.2.2.1	__median	16
2.2.2.2	__median	17
2.2.2.3	_Find_first	17
2.2.2.4	_Find_next	17
2.2.2.5	_Unchecked_flip	18
2.2.2.6	_Unchecked_reset	18
2.2.2.7	_Unchecked_set	18
2.2.2.8	_Unchecked_set	18
2.2.2.9	_Unchecked_test	19
2.2.2.10	compose1	19
2.2.2.11	compose2	19
2.2.2.12	constant0	19
2.2.2.13	constant1	19
2.2.2.14	constant2	19
2.2.2.15	copy_n	20

2.2.2.16	distance	20
2.2.2.17	identity_element	21
2.2.2.18	identity_element	21
2.2.2.19	iota	21
2.2.2.20	is_heap	21
2.2.2.21	is_heap	21
2.2.2.22	is_sorted	22
2.2.2.23	is_sorted	22
2.2.2.24	lexicographical_compare_3way	22
2.2.2.25	power	23
2.2.2.26	power	23
2.2.2.27	random_sample	23
2.2.2.28	random_sample	24
2.2.2.29	random_sample_n	24
2.2.2.30	random_sample_n	24
2.2.2.31	uninitialized_copy_n	25
2.3	Containers	26
2.3.1	Detailed Description	26
2.4	Sequences	27
2.4.1	Detailed Description	28
2.5	Associative	29
2.5.1	Detailed Description	29
2.6	Unordered Associative	30
2.6.1	Detailed Description	30
2.7	Diagnostics	31
2.7.1	Detailed Description	31
2.8	Concurrency	32
2.8.1	Detailed Description	32
2.9	Exceptions	33
2.9.1	Detailed Description	35
2.9.2	Typedef Documentation	35

2.9.2.1	terminate_handler	35
2.9.2.2	unexpected_handler	36
2.9.3	Function Documentation	36
2.9.3.1	__verbose_terminate_handler	36
2.9.3.2	copy_exception	36
2.9.3.3	current_exception	36
2.9.3.4	make_exception_ptr	36
2.9.3.5	rethrow_exception	37
2.9.3.6	rethrow_if_nested	37
2.9.3.7	rethrow_if_nested	37
2.9.3.8	set_terminate	37
2.9.3.9	set_unexpected	37
2.9.3.10	terminate	37
2.9.3.11	throw_with_nested	37
2.9.3.12	uncaught_exception	38
2.9.3.13	unexpected	38
2.10	Time	38
2.10.1	Detailed Description	38
2.11	Complex Numbers	39
2.11.1	Detailed Description	44
2.11.2	Function Documentation	44
2.11.2.1	abs	44
2.11.2.2	acos	44
2.11.2.3	acosh	44
2.11.2.4	arg	44
2.11.2.5	arg	44
2.11.2.6	asin	45
2.11.2.7	asinh	45
2.11.2.8	atan	45
2.11.2.9	atanh	45
2.11.2.10	conj	45

2.11.2.11 cos	45
2.11.2.12 cosh	46
2.11.2.13 exp	46
2.11.2.14 fabs	46
2.11.2.15 log	46
2.11.2.16 log10	46
2.11.2.17 norm	47
2.11.2.18 operator!=	47
2.11.2.19 operator!=	47
2.11.2.20 operator!=	47
2.11.2.21 operator*	47
2.11.2.22 operator*	47
2.11.2.23 operator*	48
2.11.2.24 operator*=	48
2.11.2.25 operator*=	48
2.11.2.26 operator+	48
2.11.2.27 operator+	48
2.11.2.28 operator+	48
2.11.2.29 operator+	49
2.11.2.30 operator+=	49
2.11.2.31 operator-	49
2.11.2.32 operator-	49
2.11.2.33 operator-	49
2.11.2.34 operator-	49
2.11.2.35 operator-=	50
2.11.2.36 operator/	50
2.11.2.37 operator/	50
2.11.2.38 operator/	50
2.11.2.39 operator/=	50
2.11.2.40 operator/=	50
2.11.2.41 operator<<	51

2.11.2.42	operator=	51
2.11.2.43	operator=	51
2.11.2.44	operator==	51
2.11.2.45	operator==	51
2.11.2.46	operator==	52
2.11.2.47	operator>>	52
2.11.2.48	polar	52
2.11.2.49	pow	52
2.11.2.50	pow	52
2.11.2.51	pow	53
2.11.2.52	sin	53
2.11.2.53	sinh	53
2.11.2.54	sqrt	53
2.11.2.55	tan	53
2.11.2.56	tanh	53
2.12	Condition Variables	54
2.12.1	Detailed Description	54
2.12.2	Enumeration Type Documentation	54
2.12.2.1	cv_status	54
2.13	Futures	55
2.13.1	Detailed Description	57
2.13.2	Enumeration Type Documentation	57
2.13.2.1	future_errc	57
2.13.3	Variable Documentation	57
2.13.3.1	future_category	57
2.14	I/O	57
2.14.1	Detailed Description	60
2.14.2	Typedef Documentation	61
2.14.2.1	filebuf	61
2.14.2.2	fstream	61
2.14.2.3	ifstream	61

2.14.2.4	ios	61
2.14.2.5	iostream	61
2.14.2.6	istream	61
2.14.2.7	istreamstream	61
2.14.2.8	ofstream	62
2.14.2.9	ostream	62
2.14.2.10	ostreamstream	62
2.14.2.11	streambuf	62
2.14.2.12	stringbuf	62
2.14.2.13	stringstream	62
2.14.2.14	wfilebuf	62
2.14.2.15	wfstream	63
2.14.2.16	wfstream	63
2.14.2.17	wios	63
2.14.2.18	wiostream	63
2.14.2.19	wistream	63
2.14.2.20	wistreamstream	63
2.14.2.21	wofstream	63
2.14.2.22	wostream	64
2.14.2.23	wostringstream	64
2.14.2.24	wstreambuf	64
2.14.2.25	wstringbuf	64
2.14.2.26	wstringstream	64
2.15	Memory	65
2.15.1	Detailed Description	65
2.16	Pointer Abstractions	65
2.16.1	Detailed Description	68
2.16.2	Function Documentation	68
2.16.2.1	allocate_shared	68
2.16.2.2	get_deleter	68
2.16.2.3	make_shared	68

2.16.2.4	operator<<	69
2.17	Mutexes	69
2.17.1	Detailed Description	71
2.17.2	Function Documentation	71
2.17.2.1	call_once	71
2.17.2.2	lock	71
2.17.2.3	try_lock	71
2.18	Numerics	72
2.18.1	Detailed Description	73
2.19	Rational Arithmetic	73
2.19.1	Detailed Description	74
2.20	Threads	75
2.20.1	Detailed Description	75
2.21	Utilities	76
2.21.1	Detailed Description	76
2.22	Numeric Arrays	77
2.22.1	Detailed Description	87
2.22.2	Function Documentation	87
2.22.2.1	gslice	87
2.22.2.2	gslice	87
2.22.2.3	gslice	87
2.22.2.4	gslice_array	87
2.22.2.5	indirect_array	88
2.22.2.6	mask_array	88
2.22.2.7	slice	88
2.22.2.8	slice	88
2.22.2.9	slice_array	88
2.22.2.10	valarray	88
2.22.2.11	valarray	89
2.22.2.12	valarray	89
2.22.2.13	valarray	89

2.22.2.14 valarray	89
2.22.2.15 valarray	89
2.22.2.16 valarray	89
2.22.2.17 valarray	90
2.22.2.18 valarray	90
2.22.2.19 ~gslice	90
2.22.2.20 apply	90
2.22.2.21 apply	90
2.22.2.22 cshift	91
2.22.2.23 max	91
2.22.2.24 min	91
2.22.2.25 operator!	92
2.22.2.26 operator%=	92
2.22.2.27 operator%=	92
2.22.2.28 operator%=	92
2.22.2.29 operator&=	92
2.22.2.30 operator&=	92
2.22.2.31 operator&=	93
2.22.2.32 operator*=	93
2.22.2.33 operator*=	93
2.22.2.34 operator*=	93
2.22.2.35 operator+	93
2.22.2.36 operator+=	93
2.22.2.37 operator+=	94
2.22.2.38 operator+=	94
2.22.2.39 operator-	94
2.22.2.40 operator-=	94
2.22.2.41 operator-=	94
2.22.2.42 operator-=	95
2.22.2.43 operator/=	95
2.22.2.44 operator/=	95

2.22.2.45 operator/=	95
2.22.2.46 operator<=<=	95
2.22.2.47 operator<=<=	95
2.22.2.48 operator<=<=	96
2.22.2.49 operator=.	96
2.22.2.50 operator=.	96
2.22.2.51 operator=.	96
2.22.2.52 operator=.	96
2.22.2.53 operator=.	97
2.22.2.54 operator=.	97
2.22.2.55 operator=.	97
2.22.2.56 operator=.	97
2.22.2.57 operator=.	98
2.22.2.58 operator=.	98
2.22.2.59 operator=.	98
2.22.2.60 operator=.	98
2.22.2.61 operator=.	98
2.22.2.62 operator=.	99
2.22.2.63 operator=.	99
2.22.2.64 operator=.	99
2.22.2.65 operator=.	99
2.22.2.66 operator=.	100
2.22.2.67 operator=.	100
2.22.2.68 operator>>=	100
2.22.2.69 operator>>=	100
2.22.2.70 operator>>=	100
2.22.2.71 operator[]	101
2.22.2.72 operator[]	101
2.22.2.73 operator[]	101
2.22.2.74 operator[]	102
2.22.2.75 operator[]	102

2.22.2.76 operator[]	103
2.22.2.77 operator[]	103
2.22.2.78 operator[]	104
2.22.2.79 operator[]	104
2.22.2.80 operator^=	104
2.22.2.81 operator^=	105
2.22.2.82 operator^=	105
2.22.2.83 operator =	105
2.22.2.84 operator =	105
2.22.2.85 operator =	105
2.22.2.86 operator~	105
2.22.2.87 resize	106
2.22.2.88 shift	106
2.22.2.89 size	106
2.22.2.90 size	107
2.22.2.91 size	107
2.22.2.92 start	107
2.22.2.93 start	107
2.22.2.94 stride	107
2.22.2.95 stride	107
2.22.2.96 sum	107
2.23 Mathematical Special Functions	108
2.23.1 Detailed Description	111
2.23.2 Function Documentation	111
2.23.2.1 assoc_laguerre	111
2.23.2.2 assoc_legendre	111
2.23.2.3 beta	111
2.23.2.4 comp_ellint_1	112
2.23.2.5 comp_ellint_2	112
2.23.2.6 comp_ellint_3	112
2.23.2.7 conf_hyperg	112

2.23.2.8	cyl_bessel_i	112
2.23.2.9	cyl_bessel_j	112
2.23.2.10	cyl_bessel_k	113
2.23.2.11	cyl_neumann	113
2.23.2.12	ellint_1	113
2.23.2.13	ellint_2	113
2.23.2.14	ellint_3	113
2.23.2.15	expint	113
2.23.2.16	hermite	114
2.23.2.17	hyperg	114
2.23.2.18	laguerre	114
2.23.2.19	legendre	114
2.23.2.20	riemann_zeta	114
2.23.2.21	sph_bessel	114
2.23.2.22	sph_legendre	115
2.23.2.23	sph_neumann	115
2.24	Type Traits	115
2.24.1	Detailed Description	121
2.24.2	Typedef Documentation	121
2.24.2.1	false_type	121
2.24.2.2	true_type	121
2.25	Decimal Floating-Point Arithmetic	121
2.25.1	Detailed Description	122
2.26	Binder Classes	122
2.26.1	Detailed Description	123
2.26.2	Function Documentation	124
2.26.2.1	bind	124
2.26.2.2	bind1st	124
2.26.2.3	bind2nd	124
2.26.3	Variable Documentation	124
2.26.3.1	_GLIBCXX_DEPRECATED_ATTR	124

2.27 Algorithms	125
2.27.1 Detailed Description	125
2.28 Mutating	126
2.28.1 Function Documentation	129
2.28.1.1 copy	129
2.28.1.2 copy_backward	129
2.28.1.3 copy_if	130
2.28.1.4 copy_n	130
2.28.1.5 fill	131
2.28.1.6 fill_n	131
2.28.1.7 generate	132
2.28.1.8 generate_n	132
2.28.1.9 is_partitioned	133
2.28.1.10 iter_swap	133
2.28.1.11 move	134
2.28.1.12 move	134
2.28.1.13 move_backward	135
2.28.1.14 partition	135
2.28.1.15 partition_copy	136
2.28.1.16 partition_point	136
2.28.1.17 random_shuffle	137
2.28.1.18 random_shuffle	137
2.28.1.19 remove	138
2.28.1.20 remove_copy	138
2.28.1.21 remove_copy_if	139
2.28.1.22 remove_if	139
2.28.1.23 replace	140
2.28.1.24 replace_copy_if	140
2.28.1.25 replace_if	141
2.28.1.26 reverse	141
2.28.1.27 reverse_copy	142

2.28.1.28 rotate	142
2.28.1.29 rotate_copy	143
2.28.1.30 shuffle	144
2.28.1.31 stable_partition	144
2.28.1.32 swap	145
2.28.1.33 swap_ranges	145
2.28.1.34 transform	146
2.28.1.35 transform	146
2.28.1.36 unique	147
2.28.1.37 unique	147
2.28.1.38 unique_copy	148
2.28.1.39 unique_copy	148
2.29 Non-Mutating	149
2.29.1 Function Documentation	151
2.29.1.1 adjacent_find	151
2.29.1.2 adjacent_find	151
2.29.1.3 all_of	152
2.29.1.4 any_of	152
2.29.1.5 count	153
2.29.1.6 count_if	153
2.29.1.7 equal	154
2.29.1.8 equal	154
2.29.1.9 find	155
2.29.1.10 find_end	155
2.29.1.11 find_end	156
2.29.1.12 find_first_of	157
2.29.1.13 find_first_of	157
2.29.1.14 find_if	158
2.29.1.15 find_if_not	158
2.29.1.16 for_each	159
2.29.1.17 mismatch	159

2.29.1.18 mismatch	160
2.29.1.19 none_of	160
2.29.1.20 search	161
2.29.1.21 search	161
2.29.1.22 search_n	162
2.29.1.23 search_n	163
2.30 Sorting	164
2.30.1 Function Documentation	167
2.30.1.1 inplace_merge	167
2.30.1.2 inplace_merge	167
2.30.1.3 is_sorted	168
2.30.1.4 is_sorted	169
2.30.1.5 is_sorted_until	169
2.30.1.6 is_sorted_until	169
2.30.1.7 lexicographical_compare	170
2.30.1.8 lexicographical_compare	170
2.30.1.9 max	171
2.30.1.10 max	171
2.30.1.11 max_element	172
2.30.1.12 max_element	172
2.30.1.13 merge	173
2.30.1.14 merge	173
2.30.1.15 min	174
2.30.1.16 min	174
2.30.1.17 min_element	175
2.30.1.18 min_element	175
2.30.1.19 minmax	176
2.30.1.20 minmax	176
2.30.1.21 minmax_element	177
2.30.1.22 minmax_element	177
2.30.1.23 next_permutation	178

2.30.1.24	next_permutation	178
2.30.1.25	nth_element	179
2.30.1.26	nth_element	179
2.30.1.27	partial_sort	180
2.30.1.28	partial_sort	180
2.30.1.29	partial_sort_copy	181
2.30.1.30	partial_sort_copy	182
2.30.1.31	prev_permutation	182
2.30.1.32	prev_permutation	183
2.30.1.33	sort	183
2.30.1.34	sort	184
2.30.1.35	stable_sort	184
2.30.1.36	stable_sort	185
2.31	Set Operation	186
2.31.1	Detailed Description	187
2.31.2	Function Documentation	187
2.31.2.1	includes	187
2.31.2.2	includes	188
2.31.2.3	set_difference	188
2.31.2.4	set_difference	189
2.31.2.5	set_intersection	190
2.31.2.6	set_intersection	190
2.31.2.7	set_symmetric_difference	191
2.31.2.8	set_symmetric_difference	191
2.31.2.9	set_union	192
2.31.2.10	set_union	193
2.32	Binary Search	193
2.32.1	Detailed Description	194
2.32.2	Function Documentation	195
2.32.2.1	binary_search	195
2.32.2.2	binary_search	195

2.32.2.3	equal_range	196
2.32.2.4	equal_range	196
2.32.2.5	lower_bound	197
2.32.2.6	lower_bound	198
2.32.2.7	upper_bound	198
2.32.2.8	upper_bound	199
2.33	Allocators	199
2.33.1	Detailed Description	201
2.34	Atomics	201
2.34.1	Detailed Description	207
2.34.2	Define Documentation	207
2.34.2.1	_GLIBCXX_ATOMIC_PROPERTY	207
2.34.3	Typedef Documentation	207
2.34.3.1	atomic_char	207
2.34.3.2	atomic_char16_t	207
2.34.3.3	atomic_char32_t	207
2.34.3.4	atomic_int	208
2.34.3.5	atomic_llong	208
2.34.3.6	atomic_long	208
2.34.3.7	atomic_schar	208
2.34.3.8	atomic_short	208
2.34.3.9	atomic_uchar	208
2.34.3.10	atomic_uint	208
2.34.3.11	atomic_ullong	209
2.34.3.12	atomic_ulong	209
2.34.3.13	atomic_ushort	209
2.34.3.14	atomic_wchar_t	209
2.34.3.15	memory_order	209
2.34.4	Enumeration Type Documentation	209
2.34.4.1	memory_order	209
2.34.5	Function Documentation	209

2.34.5.1	kill_dependency	209
2.35	Hashes	210
2.35.1	Detailed Description	210
2.36	Locales	210
2.36.1	Detailed Description	213
2.37	Random Number Generation	213
2.37.1	Detailed Description	214
2.37.2	Function Documentation	214
2.37.2.1	generate_canonical	214
2.38	Regular Expressions	214
2.38.1	Detailed Description	220
2.38.2	Typedef Documentation	220
2.38.2.1	cregex_token_iterator	220
2.38.2.2	csub_match	220
2.38.2.3	regex	221
2.38.2.4	sregex_token_iterator	221
2.38.2.5	ssub_match	221
2.38.2.6	wcregex_token_iterator	221
2.38.2.7	wsub_match	221
2.38.2.8	wregex	221
2.38.2.9	wsregex_token_iterator	221
2.38.2.10	wssub_match	222
2.38.3	Function Documentation	222
2.38.3.1	isctype	222
2.38.3.2	operator!=	222
2.38.3.3	operator!=	223
2.38.3.4	operator!=	223
2.38.3.5	operator!=	223
2.38.3.6	operator!=	224
2.38.3.7	operator!=	224
2.38.3.8	operator!=	225

2.38.3.9 operator!=	225
2.38.3.10 operator<	225
2.38.3.11 operator<	226
2.38.3.12 operator<	226
2.38.3.13 operator<	227
2.38.3.14 operator<	227
2.38.3.15 operator<	227
2.38.3.16 operator<	228
2.38.3.17 operator<<	228
2.38.3.18 operator<=	229
2.38.3.19 operator<=	229
2.38.3.20 operator<=	229
2.38.3.21 operator<=	230
2.38.3.22 operator<=	230
2.38.3.23 operator<=	230
2.38.3.24 operator<=	231
2.38.3.25 operator==	231
2.38.3.26 operator==	232
2.38.3.27 operator==	232
2.38.3.28 operator==	232
2.38.3.29 operator==	233
2.38.3.30 operator==	233
2.38.3.31 operator==	234
2.38.3.32 operator==	234
2.38.3.33 operator>	234
2.38.3.34 operator>	235
2.38.3.35 operator>	235
2.38.3.36 operator>	236
2.38.3.37 operator>	236
2.38.3.38 operator>	236
2.38.3.39 operator>	237

2.38.3.40 operator>=	237
2.38.3.41 operator>=	238
2.38.3.42 operator>=	238
2.38.3.43 operator>=	238
2.38.3.44 operator>=	239
2.38.3.45 operator>=	239
2.38.3.46 operator>=	240
2.38.3.47 regex_match	240
2.38.3.48 regex_match	241
2.38.3.49 regex_match	241
2.38.3.50 regex_match	242
2.38.3.51 regex_match	243
2.38.3.52 regex_match	243
2.38.3.53 regex_replace	244
2.38.3.54 regex_replace	245
2.38.3.55 regex_search	245
2.38.3.56 regex_search	246
2.38.3.57 regex_search	247
2.38.3.58 regex_search	247
2.38.3.59 regex_search	248
2.38.3.60 regex_search	249
2.38.3.61 swap	250
2.38.3.62 swap	250
2.38.3.63 value	250
2.39 Function Objects	251
2.39.1 Detailed Description	252
2.39.2 Function Documentation	253
2.39.2.1 mem_fn	253
2.40 Arithmetic Classes	253
2.40.1 Detailed Description	254
2.41 Comparison Classes	254

2.41.1 Detailed Description	255
2.42 Boolean Operations Classes	255
2.42.1 Detailed Description	255
2.43 Negators	256
2.43.1 Detailed Description	256
2.43.2 Function Documentation	257
2.43.2.1 not1	257
2.43.2.2 not2	257
2.44 Adaptors for pointers to functions	257
2.44.1 Detailed Description	258
2.44.2 Function Documentation	258
2.44.2.1 ptr_fun	258
2.44.2.2 ptr_fun	259
2.45 Adaptors for pointers to members	259
2.45.1 Detailed Description	260
2.46 Heap	260
2.46.1 Function Documentation	262
2.46.1.1 is_heap	262
2.46.1.2 is_heap	262
2.46.1.3 is_heap_until	262
2.46.1.4 is_heap_until	263
2.46.1.5 make_heap	263
2.46.1.6 make_heap	264
2.46.1.7 pop_heap	264
2.46.1.8 pop_heap	265
2.46.1.9 push_heap	265
2.46.1.10 push_heap	265
2.46.1.11 sort_heap	266
2.46.1.12 sort_heap	266
2.47 Iterators	267
2.47.1 Detailed Description	271

2.47.2	Function Documentation	271
2.47.2.1	__iterator_category	271
2.47.2.2	back_inserter	271
2.47.2.3	front_inserter	272
2.47.2.4	inserter	272
2.47.2.5	operator!=	272
2.47.2.6	operator==	273
2.47.2.7	operator==	273
2.48	Strings	273
2.48.1	Typedef Documentation	274
2.48.1.1	string	274
2.48.1.2	u16string	274
2.48.1.3	u32string	274
2.48.1.4	wstring	274
2.49	Policy-Based Data Structures	275
2.49.1	Detailed Description	276
2.50	Random Number Generators	276
2.50.1	Detailed Description	278
2.50.2	Typedef Documentation	279
2.50.2.1	minstd_rand	279
2.50.2.2	minstd_rand0	279
2.50.2.3	mt19937	279
2.50.2.4	mt19937_64	279
2.50.3	Function Documentation	280
2.50.3.1	operator!=	280
2.50.3.2	operator!=	280
2.50.3.3	operator!=	281
2.50.3.4	operator!=	281
2.50.3.5	operator!=	282
2.50.3.6	operator!=	282
2.50.3.7	operator<<	283

2.51 Random Number Distributions	283
2.52 Uniform	284
2.52.1 Function Documentation	285
2.52.1.1 operator!=	285
2.52.1.2 operator!=	285
2.52.1.3 operator<<	285
2.52.1.4 operator<<	286
2.52.1.5 operator==	286
2.52.1.6 operator==	286
2.52.1.7 operator>>	287
2.52.1.8 operator>>	287
2.53 Normal	288
2.53.1 Function Documentation	289
2.53.1.1 operator!=	289
2.53.1.2 operator!=	289
2.53.1.3 operator!=	290
2.53.1.4 operator!=	290
2.53.1.5 operator!=	290
2.53.1.6 operator!=	290
2.53.1.7 operator!=	290
2.53.1.8 operator<<	291
2.53.1.9 operator==	291
2.53.1.10 operator>>	291
2.54 Bernoulli	292
2.54.1 Function Documentation	293
2.54.1.1 operator!=	293
2.54.1.2 operator!=	293
2.54.1.3 operator!=	294
2.54.1.4 operator!=	294
2.54.1.5 operator<<	294
2.54.1.6 operator<<	294

2.54.1.7	operator==	295
2.54.1.8	operator==	295
2.54.1.9	operator>>	295
2.54.1.10	operator>>	296
2.55	Poisson	296
2.55.1	Function Documentation	299
2.55.1.1	operator!=	299
2.55.1.2	operator!=	299
2.55.1.3	operator!=	299
2.55.1.4	operator!=	299
2.55.1.5	operator!=	299
2.55.1.6	operator!=	300
2.55.1.7	operator!=	300
2.55.1.8	operator<<	300
2.55.1.9	operator<<	300
2.55.1.10	operator<<	301
2.55.1.11	operator==	301
2.55.1.12	operator==	302
2.55.1.13	operator==	302
2.55.1.14	operator==	302
2.55.1.15	operator==	302
2.55.1.16	operator==	303
2.55.1.17	operator>>	303
2.55.1.18	operator>>	303
2.55.1.19	operator>>	304
2.56	Random Number Utilities	304
3	Directory Documentation	307
3.1	include/backward/ Directory Reference	308
3.2	include/x86_64-unknown-linux-gnu/bits/ Directory Reference	309
3.3	include/bits/ Directory Reference	311

3.4	include/debug/ Directory Reference	314
3.5	include/decimal/ Directory Reference	315
3.6	include/ext/pb_ds/detail/ Directory Reference	316
3.7	/mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/ Directory Reference	317
3.8	/mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/ Directory Reference	318
3.9	include/ext/ Directory Reference	319
3.10	/mnt/share/src/gcc.svn-trunk/ Directory Reference	321
3.11	include/profile/impl/ Directory Reference	322
3.12	include/ Directory Reference	324
3.13	/mnt/share/src/gcc.svn-trunk/libstdc++-v3/ Directory Reference	327
3.14	/mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/ Directory Reference	328
3.15	include/parallel/ Directory Reference	329
3.16	include/ext/pb_ds/ Directory Reference	332
3.17	include/profile/ Directory Reference	334
3.18	/mnt/share/src/ Directory Reference	335
3.19	include/tr1/ Directory Reference	336
3.20	include/tr1_impl/ Directory Reference	337
3.21	include/x86_64-unknown-linux-gnu/ Directory Reference	338
4	Namespace Documentation	339
4.1	__gnu_cxx Namespace Reference	339
4.1.1	Detailed Description	361
4.1.2	Function Documentation	361
4.1.2.1	__static_pointer_cast	361
4.1.2.2	__static_pointer_cast	361
4.1.2.3	_Bit_scan_forward	362
4.1.2.4	operator!=	362
4.1.2.5	operator!=	362
4.1.2.6	operator!=	363
4.1.2.7	operator+	363

4.1.2.8	operator+	364
4.1.2.9	operator+	364
4.1.2.10	operator+	365
4.1.2.11	operator+	365
4.1.2.12	operator<	366
4.1.2.13	operator<	366
4.1.2.14	operator<	366
4.1.2.15	operator<=	367
4.1.2.16	operator<=	367
4.1.2.17	operator<=	368
4.1.2.18	operator==	368
4.1.2.19	operator==	368
4.1.2.20	operator==	369
4.1.2.21	operator==	369
4.1.2.22	operator>	370
4.1.2.23	operator>	370
4.1.2.24	operator>	371
4.1.2.25	operator>=	371
4.1.2.26	operator>=	372
4.1.2.27	operator>=	372
4.1.2.28	swap	373
4.2	<code>__gnu_cxx::__detail</code> Namespace Reference	373
4.2.1	Detailed Description	374
4.2.2	Function Documentation	374
4.2.2.1	<code>__bit_allocate</code>	374
4.2.2.2	<code>__bit_free</code>	374
4.2.2.3	<code>__num_bitmaps</code>	374
4.2.2.4	<code>__num_blocks</code>	375
4.3	<code>__gnu_cxx::typelist</code> Namespace Reference	375
4.3.1	Detailed Description	375
4.3.2	Function Documentation	375

4.3.2.1	<code>apply_generator</code>	375
4.4	<code>__gnu_debug</code> Namespace Reference	376
4.4.1	Detailed Description	382
4.4.2	Function Documentation	382
4.4.2.1	<code>__check_dereferenceable</code>	382
4.4.2.2	<code>__check_dereferenceable</code>	382
4.4.2.3	<code>__check_dereferenceable</code>	383
4.4.2.4	<code>__check_singular</code>	383
4.4.2.5	<code>__check_singular</code>	383
4.4.2.6	<code>__check_singular_aux</code>	383
4.4.2.7	<code>__check_string</code>	383
4.4.2.8	<code>__check_string</code>	384
4.4.2.9	<code>__valid_range</code>	384
4.4.2.10	<code>__valid_range</code>	384
4.4.2.11	<code>__valid_range_aux</code>	384
4.4.2.12	<code>__valid_range_aux</code>	384
4.4.2.13	<code>__valid_range_aux2</code>	385
4.4.2.14	<code>__valid_range_aux2</code>	385
4.5	<code>__gnu_internal</code> Namespace Reference	385
4.5.1	Detailed Description	385
4.6	<code>__gnu_parallel</code> Namespace Reference	385
4.6.1	Detailed Description	402
4.6.2	Typedef Documentation	402
4.6.2.1	<code>_BinIndex</code>	402
4.6.2.2	<code>_CASable</code>	402
4.6.2.3	<code>_SequenceIndex</code>	402
4.6.2.4	<code>_ThreadIndex</code>	402
4.6.3	Enumeration Type Documentation	403
4.6.3.1	<code>_AlgorithmStrategy</code>	403
4.6.3.2	<code>_FindAlgorithm</code>	403
4.6.3.3	<code>_MultiwayMergeAlgorithm</code>	403

4.6.3.4	_Parallelism	403
4.6.3.5	_PartialSumAlgorithm	403
4.6.3.6	_SortAlgorithm	404
4.6.3.7	_SplittingAlgorithm	404
4.6.4	Function Documentation	404
4.6.4.1	__calc_borders	404
4.6.4.2	__compare_and_swap	404
4.6.4.3	__compare_and_swap_32	405
4.6.4.4	__compare_and_swap_64	405
4.6.4.5	__decode2	405
4.6.4.6	__determine_samples	406
4.6.4.7	__encode2	406
4.6.4.8	__fetch_and_add	407
4.6.4.9	__fetch_and_add_32	407
4.6.4.10	__fetch_and_add_64	408
4.6.4.11	__find_template	408
4.6.4.12	__find_template	409
4.6.4.13	__find_template	409
4.6.4.14	__find_template	410
4.6.4.15	__for_each_template_random_access	411
4.6.4.16	__for_each_template_random_access_ed	411
4.6.4.17	__for_each_template_random_access_omp_loop	412
4.6.4.18	__for_each_template_random_access_omp_loop_-static	413
4.6.4.19	__for_each_template_random_access_workstealing	414
4.6.4.20	__is_sorted	414
4.6.4.21	__median_of_three_iterators	415
4.6.4.22	__merge_advance	415
4.6.4.23	__merge_advance_movc	416
4.6.4.24	__merge_advance_usual	417
4.6.4.25	__parallel_merge_advance	417

4.6.4.26	<code>__parallel_merge_advance</code>	418
4.6.4.27	<code>__parallel_nth_element</code>	419
4.6.4.28	<code>__parallel_partial_sort</code>	419
4.6.4.29	<code>__parallel_partial_sum</code>	419
4.6.4.30	<code>__parallel_partial_sum_basecase</code>	420
4.6.4.31	<code>__parallel_partial_sum_linear</code>	421
4.6.4.32	<code>__parallel_partition</code>	421
4.6.4.33	<code>__parallel_random_shuffle</code>	422
4.6.4.34	<code>__parallel_random_shuffle_drs</code>	422
4.6.4.35	<code>__parallel_random_shuffle_drs_pu</code>	423
4.6.4.36	<code>__parallel_sort</code>	423
4.6.4.37	<code>__parallel_sort</code>	424
4.6.4.38	<code>__parallel_sort</code>	425
4.6.4.39	<code>__parallel_sort</code>	426
4.6.4.40	<code>__parallel_sort</code>	426
4.6.4.41	<code>__parallel_sort</code>	427
4.6.4.42	<code>__parallel_sort</code>	428
4.6.4.43	<code>__parallel_sort_qs</code>	428
4.6.4.44	<code>__parallel_sort_qs_conquer</code>	429
4.6.4.45	<code>__parallel_sort_qs_divide</code>	429
4.6.4.46	<code>__parallel_sort_qsb</code>	430
4.6.4.47	<code>__parallel_unique_copy</code>	430
4.6.4.48	<code>__parallel_unique_copy</code>	430
4.6.4.49	<code>__qsb_conquer</code>	431
4.6.4.50	<code>__qsb_divide</code>	432
4.6.4.51	<code>__qsb_local_sort_with_helping</code>	432
4.6.4.52	<code>__random_number_pow2</code>	433
4.6.4.53	<code>__rd_log2</code>	433
4.6.4.54	<code>__round_up_to_pow2</code>	433
4.6.4.55	<code>__search_template</code>	434
4.6.4.56	<code>__sequential_multiway_merge</code>	434

4.6.4.57	__sequential_random_shuffle	435
4.6.4.58	__shrink	435
4.6.4.59	__shrink_and_double	436
4.6.4.60	__yield	436
4.6.4.61	equally_split	436
4.6.4.62	equally_split_point	437
4.6.4.63	list_partition	437
4.6.4.64	max	438
4.6.4.65	min	438
4.6.4.66	multiseq_partition	439
4.6.4.67	multiseq_selection	439
4.6.4.68	multiway_merge	440
4.6.4.69	multiway_merge_3_variant	442
4.6.4.70	multiway_merge_4_variant	443
4.6.4.71	multiway_merge_exact_splitting	443
4.6.4.72	multiway_merge_loser_tree	444
4.6.4.73	multiway_merge_loser_tree_sentinel	445
4.6.4.74	multiway_merge_loser_tree_unguarded	445
4.6.4.75	multiway_merge_sampling_splitting	446
4.6.4.76	multiway_merge_sentinels	446
4.6.4.77	parallel_multiway_merge	448
4.6.4.78	parallel_sort_mwms	449
4.6.4.79	parallel_sort_mwms_pu	450
4.6.5	Variable Documentation	450
4.6.5.1	_CASable_bits	450
4.6.5.2	_CASable_mask	450
4.7	__gnu_pbds Namespace Reference	450
4.7.1	Detailed Description	453
4.8	__gnu_profile Namespace Reference	453
4.8.1	Detailed Description	458
4.8.2	Typedef Documentation	459

4.8.2.1	<code>__env_t</code>	459
4.8.3	Function Documentation	459
4.8.3.1	<code>__get__global_lock</code>	459
4.8.3.2	<code>__profcxx_init</code>	459
4.8.3.3	<code>__report</code>	459
4.9	<code>__gnu_sequential</code> Namespace Reference	460
4.9.1	Detailed Description	460
4.10	<code>abi</code> Namespace Reference	460
4.10.1	Detailed Description	460
4.11	<code>std</code> Namespace Reference	460
4.11.1	Detailed Description	598
4.11.2	Typedef Documentation	599
4.11.2.1	<code>new_handler</code>	599
4.11.2.2	<code>streamoff</code>	599
4.11.2.3	<code>streampos</code>	599
4.11.2.4	<code>streamsize</code>	599
4.11.2.5	<code>u16streampos</code>	599
4.11.2.6	<code>u32streampos</code>	599
4.11.2.7	<code>wstreampos</code>	599
4.11.3	Enumeration Type Documentation	600
4.11.3.1	<code>"@53</code>	600
4.11.3.2	<code>float_denorm_style</code>	600
4.11.3.3	<code>float_round_style</code>	600
4.11.4	Function Documentation	601
4.11.4.1	<code>__final_insertion_sort</code>	601
4.11.4.2	<code>__final_insertion_sort</code>	601
4.11.4.3	<code>__find</code>	601
4.11.4.4	<code>__find</code>	601
4.11.4.5	<code>__find_if</code>	602
4.11.4.6	<code>__find_if</code>	602
4.11.4.7	<code>__find_if_not</code>	602

4.11.4.8	<code>__find_if_not</code>	602
4.11.4.9	<code>__gcd</code>	602
4.11.4.10	<code>__heap_select</code>	603
4.11.4.11	<code>__heap_select</code>	603
4.11.4.12	<code>__inplace_stable_partition</code>	603
4.11.4.13	<code>__inplace_stable_sort</code>	603
4.11.4.14	<code>__inplace_stable_sort</code>	604
4.11.4.15	<code>__insertion_sort</code>	604
4.11.4.16	<code>__insertion_sort</code>	604
4.11.4.17	<code>__introsort_loop</code>	604
4.11.4.18	<code>__introsort_loop</code>	605
4.11.4.19	<code>__lg</code>	605
4.11.4.20	<code>__merge_adaptive</code>	605
4.11.4.21	<code>__merge_adaptive</code>	605
4.11.4.22	<code>__merge_backward</code>	606
4.11.4.23	<code>__merge_backward</code>	606
4.11.4.24	<code>__merge_without_buffer</code>	606
4.11.4.25	<code>__merge_without_buffer</code>	607
4.11.4.26	<code>__move_median_first</code>	607
4.11.4.27	<code>__move_median_first</code>	607
4.11.4.28	<code>__partition</code>	607
4.11.4.29	<code>__partition</code>	608
4.11.4.30	<code>__reverse</code>	608
4.11.4.31	<code>__reverse</code>	608
4.11.4.32	<code>__rotate</code>	608
4.11.4.33	<code>__rotate</code>	609
4.11.4.34	<code>__rotate</code>	609
4.11.4.35	<code>__rotate_adaptive</code>	609
4.11.4.36	<code>__search_n</code>	609
4.11.4.37	<code>__search_n</code>	610
4.11.4.38	<code>__search_n</code>	610

4.11.4.39 <code>__search_n</code>	610
4.11.4.40 <code>__stable_partition_adaptive</code>	610
4.11.4.41 <code>__unguarded_insertion_sort</code>	611
4.11.4.42 <code>__unguarded_insertion_sort</code>	611
4.11.4.43 <code>__unguarded_linear_insert</code>	611
4.11.4.44 <code>__unguarded_linear_insert</code>	611
4.11.4.45 <code>__unguarded_partition</code>	611
4.11.4.46 <code>__unguarded_partition</code>	612
4.11.4.47 <code>__unguarded_partition_pivot</code>	612
4.11.4.48 <code>__unguarded_partition_pivot</code>	612
4.11.4.49 <code>__unique_copy</code>	613
4.11.4.50 <code>__unique_copy</code>	613
4.11.4.51 <code>__unique_copy</code>	613
4.11.4.52 <code>__unique_copy</code>	613
4.11.4.53 <code>__unique_copy</code>	614
4.11.4.54 <code>__unique_copy</code>	614
4.11.4.55 <code>_Construct</code>	614
4.11.4.56 <code>_Destroy</code>	614
4.11.4.57 <code>_Destroy</code>	614
4.11.4.58 <code>accumulate</code>	615
4.11.4.59 <code>accumulate</code>	615
4.11.4.60 <code>addressof</code>	616
4.11.4.61 <code>adjacent_difference</code>	616
4.11.4.62 <code>adjacent_difference</code>	616
4.11.4.63 <code>advance</code>	617
4.11.4.64 <code>bind</code>	618
4.11.4.65 <code>boolalpha</code>	618
4.11.4.66 <code>cref</code>	618
4.11.4.67 <code>cref</code>	618
4.11.4.68 <code>dec</code>	618
4.11.4.69 <code>distance</code>	619

4.11.4.70 endl	619
4.11.4.71 ends	620
4.11.4.72 fixed	620
4.11.4.73 flush	620
4.11.4.74 forward	620
4.11.4.75 forward	620
4.11.4.76 get_money	621
4.11.4.77 get_temporary_buffer	621
4.11.4.78 getline	622
4.11.4.79 getline	622
4.11.4.80 getline	623
4.11.4.81 getline	623
4.11.4.82 has_facet	624
4.11.4.83 hex	624
4.11.4.84 inner_product	625
4.11.4.85 inner_product	625
4.11.4.86 internal	626
4.11.4.87 iota	626
4.11.4.88 isalnum	626
4.11.4.89 isalpha	626
4.11.4.90 iscntrl	627
4.11.4.91 isdigit	627
4.11.4.92 isgraph	627
4.11.4.93 islower	627
4.11.4.94 isprint	627
4.11.4.95 ispunct	627
4.11.4.96 isspace	627
4.11.4.97 isupper	627
4.11.4.98 isxdigit	628
4.11.4.99 left	628
4.11.4.100make_pair	628

4.11.4.101noboolalpha	628
4.11.4.102noshowbase	629
4.11.4.103noshowpoint	629
4.11.4.104noshowpos	629
4.11.4.105noskipws	629
4.11.4.106nounitbuf	629
4.11.4.107nouppercase	629
4.11.4.108oct	630
4.11.4.109operator!=	630
4.11.4.110operator!=	630
4.11.4.111operator!=	631
4.11.4.112operator!=	631
4.11.4.113operator!=	631
4.11.4.114operator!=	631
4.11.4.115operator!=	632
4.11.4.116operator!=	632
4.11.4.117operator!=	632
4.11.4.118operator!=	632
4.11.4.119operator!=	632
4.11.4.120operator!=	633
4.11.4.121operator!=	633
4.11.4.122operator!=	633
4.11.4.123operator!=	633
4.11.4.124operator!=	633
4.11.4.125operator!=	633
4.11.4.126operator&	634
4.11.4.127operator+	634
4.11.4.128operator+	634
4.11.4.129operator+	635
4.11.4.130operator+	635
4.11.4.131operator+	636

4.11.4.132operator<	636
4.11.4.133operator<	636
4.11.4.134operator<	637
4.11.4.135operator<	637
4.11.4.136operator<	638
4.11.4.137operator<	638
4.11.4.138operator<	639
4.11.4.139operator<	639
4.11.4.140operator<	639
4.11.4.141operator<	640
4.11.4.142operator<	640
4.11.4.143operator<	641
4.11.4.144operator<	641
4.11.4.145operator<	642
4.11.4.146operator<<	642
4.11.4.147operator<<	643
4.11.4.148operator<<	643
4.11.4.149operator<<	644
4.11.4.150operator<<	644
4.11.4.151operator<<	645
4.11.4.152operator<<	645
4.11.4.153operator<<	645
4.11.4.154operator<<	646
4.11.4.155operator<<	646
4.11.4.156operator<<	647
4.11.4.157operator<<	647
4.11.4.158operator<<	648
4.11.4.159operator<<	649
4.11.4.160operator<=	649
4.11.4.161operator<=	649
4.11.4.162operator<=	650

4.11.4.163operator<=	650
4.11.4.164operator<=	650
4.11.4.165operator<=	650
4.11.4.166operator<=	650
4.11.4.167operator<=	651
4.11.4.168operator<=	651
4.11.4.169operator<=	651
4.11.4.170operator<=	651
4.11.4.171operator<=	651
4.11.4.172operator<=	651
4.11.4.173operator<=	652
4.11.4.174operator==	652
4.11.4.175operator==	653
4.11.4.176operator==	653
4.11.4.177operator==	653
4.11.4.178operator==	654
4.11.4.179operator==	654
4.11.4.180operator==	655
4.11.4.181operator==	655
4.11.4.182operator==	656
4.11.4.183operator==	656
4.11.4.184operator==	656
4.11.4.185operator==	657
4.11.4.186operator==	657
4.11.4.187operator==	657
4.11.4.188operator==	658
4.11.4.189operator==	658
4.11.4.190operator==	658
4.11.4.191operator==	659
4.11.4.192operator>	659
4.11.4.193operator>	659

4.11.4.194operator>	660
4.11.4.195operator>	660
4.11.4.196operator>	660
4.11.4.197operator>	661
4.11.4.198operator>	661
4.11.4.199operator>	661
4.11.4.200operator>	661
4.11.4.201operator>	661
4.11.4.202operator>	662
4.11.4.203operator>	662
4.11.4.204operator>	662
4.11.4.205operator>	662
4.11.4.206operator>=	663
4.11.4.207operator>=	663
4.11.4.208operator>=	663
4.11.4.209operator>=	663
4.11.4.210operator>=	664
4.11.4.211operator>=	664
4.11.4.212operator>=	664
4.11.4.213operator>=	664
4.11.4.214operator>=	664
4.11.4.215operator>=	665
4.11.4.216operator>=	665
4.11.4.217operator>=	665
4.11.4.218operator>=	665
4.11.4.219operator>=	666
4.11.4.220operator>>	666
4.11.4.221operator>>	667
4.11.4.222operator>>	667
4.11.4.223operator>>	667
4.11.4.224operator>>	668

4.11.4.225operator>>	669
4.11.4.226operator>>	670
4.11.4.227operator>>	670
4.11.4.228operator>>	671
4.11.4.229operator>>	671
4.11.4.230operator>>	672
4.11.4.231operator^	673
4.11.4.232operator	673
4.11.4.233partial_sum	674
4.11.4.234partial_sum	674
4.11.4.235put_money	675
4.11.4.236ref	675
4.11.4.237ref	675
4.11.4.238replace_copy	675
4.11.4.239resetiosflags	676
4.11.4.240return_temporary_buffer	676
4.11.4.241right	677
4.11.4.242scientific	677
4.11.4.243set_new_handler	677
4.11.4.244setbase	677
4.11.4.245setfill	677
4.11.4.246setiosflags	678
4.11.4.247setprecision	678
4.11.4.248setw	678
4.11.4.249showbase	678
4.11.4.250showpoint	679
4.11.4.251showpos	679
4.11.4.252skipws	679
4.11.4.253swap	679
4.11.4.254swap	679
4.11.4.255swap	680

4.11.4.256	swap	680
4.11.4.257	swap	680
4.11.4.258	swap	680
4.11.4.259	swap	681
4.11.4.260	swap	681
4.11.4.261	swap	681
4.11.4.262	swap	681
4.11.4.263	swap	682
4.11.4.264	tolower	682
4.11.4.265	toupper	682
4.11.4.266	uninitialized_copy	682
4.11.4.267	uninitialized_copy_n	683
4.11.4.268	uninitialized_fill	683
4.11.4.269	uninitialized_fill_n	683
4.11.4.270	unitbuf	684
4.11.4.271	uppercase	684
4.11.4.272	use_facet	684
4.11.4.273	ws	685
4.11.5	Variable Documentation	685
4.11.5.1	__invoke	685
4.11.5.2	cerr	686
4.11.5.3	cin	686
4.11.5.4	clog	686
4.11.5.5	cout	686
4.11.5.6	wcerr	686
4.11.5.7	wcin	686
4.11.5.8	wclog	686
4.11.5.9	wcout	686
4.12	std::__debug Namespace Reference	686
4.12.1	Detailed Description	692
4.13	std::__detail Namespace Reference	692

4.13.1 Detailed Description	693
4.14 std::__parallel Namespace Reference	693
4.14.1 Detailed Description	718
4.15 std::__profile Namespace Reference	719
4.15.1 Detailed Description	725
4.16 std::chrono Namespace Reference	726
4.16.1 Detailed Description	728
4.16.2 Typedef Documentation	729
4.16.2.1 hours	729
4.16.2.2 microseconds	729
4.16.2.3 milliseconds	729
4.16.2.4 minutes	729
4.16.2.5 nanoseconds	729
4.16.2.6 seconds	729
4.16.3 Function Documentation	730
4.16.3.1 duration_cast	730
4.16.3.2 time_point_cast	730
4.17 std::decimal Namespace Reference	730
4.17.1 Detailed Description	741
4.17.2 Function Documentation	741
4.17.2.1 decimal32_to_long_long	741
4.18 std::placeholders Namespace Reference	741
4.18.1 Detailed Description	741
4.19 std::regex_constants Namespace Reference	741
4.19.1 Detailed Description	743
4.19.2 Typedef Documentation	743
4.19.2.1 match_flag_type	743
4.19.2.2 syntax_option_type	743
4.19.3 Enumeration Type Documentation	744
4.19.3.1 __match_flag	744
4.19.3.2 __syntax_option	744

4.19.3.3	<code>error_type</code>	744
4.19.4	Function Documentation	744
4.19.4.1	<code>error_backref</code>	744
4.19.4.2	<code>error_badbrace</code>	745
4.19.4.3	<code>error_badrepeat</code>	745
4.19.4.4	<code>error_brace</code>	745
4.19.4.5	<code>error_brack</code>	745
4.19.4.6	<code>error_collate</code>	745
4.19.4.7	<code>error_complexity</code>	745
4.19.4.8	<code>error_ctype</code>	745
4.19.4.9	<code>error_escape</code>	745
4.19.4.10	<code>error_paren</code>	746
4.19.4.11	<code>error_range</code>	746
4.19.4.12	<code>error_space</code>	746
4.19.4.13	<code>error_stack</code>	746
4.19.5	Variable Documentation	746
4.19.5.1	<code>awk</code>	746
4.19.5.2	<code>basic</code>	746
4.19.5.3	<code>collate</code>	747
4.19.5.4	<code>ECMAScript</code>	747
4.19.5.5	<code>egrep</code>	747
4.19.5.6	<code>extended</code>	747
4.19.5.7	<code>format_default</code>	747
4.19.5.8	<code>format_first_only</code>	748
4.19.5.9	<code>format_no_copy</code>	748
4.19.5.10	<code>format_sed</code>	748
4.19.5.11	<code>grep</code>	749
4.19.5.12	<code>icase</code>	749
4.19.5.13	<code>match_any</code>	749
4.19.5.14	<code>match_continuous</code>	749
4.19.5.15	<code>match_default</code>	749

4.19.5.16	<code>match_not_bol</code>	749
4.19.5.17	<code>match_not_bow</code>	750
4.19.5.18	<code>match_not_eol</code>	750
4.19.5.19	<code>match_not_eow</code>	750
4.19.5.20	<code>match_not_null</code>	750
4.19.5.21	<code>match_prev_avail</code>	750
4.19.5.22	<code>nosubs</code>	750
4.19.5.23	<code>optimize</code>	751
4.20	<code>std::rel_ops</code> Namespace Reference	751
4.20.1	Detailed Description	751
4.20.2	Function Documentation	751
4.20.2.1	<code>operator!=</code>	751
4.20.2.2	<code>operator<=</code>	752
4.20.2.3	<code>operator></code>	752
4.20.2.4	<code>operator>=</code>	752
4.21	<code>std::this_thread</code> Namespace Reference	753
4.21.1	Detailed Description	753
4.21.2	Function Documentation	753
4.21.2.1	<code>get_id</code>	753
4.21.2.2	<code>sleep_for</code>	753
4.21.2.3	<code>sleep_until</code>	754
4.21.2.4	<code>yield</code>	754
4.22	<code>std::tr1</code> Namespace Reference	754
4.22.1	Detailed Description	762
4.23	<code>std::tr1::__detail</code> Namespace Reference	762
4.23.1	Detailed Description	762
5	Class Documentation	763
5.1	<code>__atomic0::atomic_address</code> Struct Reference	763
5.1.1	Detailed Description	764
5.2	<code>__atomic0::atomic_bool</code> Struct Reference	764

5.2.1	Detailed Description	764
5.3	<code>__atomic0::atomic_flag</code> Struct Reference	765
5.3.1	Detailed Description	765
5.4	<code>__atomic2::atomic_address</code> Struct Reference	765
5.4.1	Detailed Description	766
5.5	<code>__atomic2::atomic_bool</code> Struct Reference	766
5.5.1	Detailed Description	767
5.6	<code>__atomic2::atomic_flag</code> Struct Reference	767
5.6.1	Detailed Description	767
5.7	<code>__cxxabiv1::__forced_unwind</code> Class Reference	767
5.7.1	Detailed Description	767
5.8	<code>__gnu_cxx::__common_pool_policy< _PoolTp, _Thread ></code> Struct Template Reference	768
5.8.1	Detailed Description	768
5.9	<code>__gnu_cxx::__detail::__mini_vector< _Tp ></code> Class Template Reference	768
5.9.1	Detailed Description	769
5.10	<code>__gnu_cxx::__detail::__Bitmap_counter< _Tp ></code> Class Template Ref- erence	769
5.10.1	Detailed Description	770
5.11	<code>__gnu_cxx::__detail::__Ffit_finder< _Tp ></code> Class Template Reference	771
5.11.1	Detailed Description	772
5.11.2	Member Typedef Documentation	772
5.11.2.1	<code>argument_type</code>	772
5.11.2.2	<code>result_type</code>	772
5.12	<code>__gnu_cxx::__mt_alloc< _Tp, _Poolp ></code> Class Template Reference	772
5.12.1	Detailed Description	774
5.13	<code>__gnu_cxx::__mt_alloc_base< _Tp ></code> Class Template Reference	774
5.13.1	Detailed Description	775
5.14	<code>__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread ></code> Struct Template Reference	775
5.14.1	Detailed Description	776
5.15	<code>__gnu_cxx::__pool< false ></code> Class Template Reference	776

5.15.1 Detailed Description	777
5.16 <code>__gnu_cxx::__pool< true ></code> Class Template Reference	777
5.16.1 Detailed Description	779
5.17 <code>__gnu_cxx::__pool_alloc< _Tp ></code> Class Template Reference	779
5.17.1 Detailed Description	781
5.18 <code>__gnu_cxx::__pool_alloc_base</code> Class Reference	781
5.18.1 Detailed Description	782
5.19 <code>__gnu_cxx::__pool_base</code> Struct Reference	782
5.19.1 Detailed Description	783
5.20 <code>__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc ></code> Class Template Reference	784
5.20.1 Detailed Description	786
5.21 <code>__gnu_cxx::__scoped_lock</code> Class Reference	787
5.21.1 Detailed Description	787
5.22 <code>__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base ></code> Class Template Reference	787
5.22.1 Detailed Description	792
5.22.2 Constructor & Destructor Documentation	792
5.22.2.1 <code>__versa_string</code>	792
5.22.2.2 <code>__versa_string</code>	792
5.22.2.3 <code>__versa_string</code>	792
5.22.2.4 <code>__versa_string</code>	793
5.22.2.5 <code>__versa_string</code>	793
5.22.2.6 <code>__versa_string</code>	793
5.22.2.7 <code>__versa_string</code>	794
5.22.2.8 <code>__versa_string</code>	794
5.22.2.9 <code>__versa_string</code>	795
5.22.2.10 <code>__versa_string</code>	795
5.22.2.11 <code>__versa_string</code>	795
5.22.2.12 <code>~__versa_string</code>	796
5.22.3 Member Function Documentation	796
5.22.3.1 <code>append</code>	796

5.22.3.2	append	796
5.22.3.3	append	797
5.22.3.4	append	797
5.22.3.5	append	798
5.22.3.6	append	798
5.22.3.7	append	799
5.22.3.8	assign	799
5.22.3.9	assign	799
5.22.3.10	assign	800
5.22.3.11	assign	800
5.22.3.12	assign	801
5.22.3.13	assign	801
5.22.3.14	assign	802
5.22.3.15	assign	802
5.22.3.16	at	803
5.22.3.17	at	803
5.22.3.18	back	804
5.22.3.19	back	804
5.22.3.20	begin	804
5.22.3.21	begin	804
5.22.3.22	c_str	805
5.22.3.23	capacity	805
5.22.3.24	cbegin	805
5.22.3.25	cend	805
5.22.3.26	clear	806
5.22.3.27	compare	806
5.22.3.28	compare	806
5.22.3.29	compare	807
5.22.3.30	compare	808
5.22.3.31	compare	809
5.22.3.32	compare	809

5.22.3.33 copy	810
5.22.3.34 crbegin	811
5.22.3.35 crend	811
5.22.3.36 data	811
5.22.3.37 empty	811
5.22.3.38 end	812
5.22.3.39 end	812
5.22.3.40 erase	812
5.22.3.41 erase	813
5.22.3.42 erase	813
5.22.3.43 find	813
5.22.3.44 find	814
5.22.3.45 find	815
5.22.3.46 find	815
5.22.3.47 find_first_not_of	816
5.22.3.48 find_first_not_of	816
5.22.3.49 find_first_not_of	817
5.22.3.50 find_first_not_of	817
5.22.3.51 find_first_of	818
5.22.3.52 find_first_of	818
5.22.3.53 find_first_of	819
5.22.3.54 find_first_of	819
5.22.3.55 find_last_not_of	820
5.22.3.56 find_last_not_of	820
5.22.3.57 find_last_not_of	821
5.22.3.58 find_last_not_of	822
5.22.3.59 find_last_of	822
5.22.3.60 find_last_of	823
5.22.3.61 find_last_of	823
5.22.3.62 find_last_of	824
5.22.3.63 front	824

5.22.3.64	front	824
5.22.3.65	get_allocator	825
5.22.3.66	insert	825
5.22.3.67	insert	825
5.22.3.68	insert	826
5.22.3.69	insert	827
5.22.3.70	insert	827
5.22.3.71	insert	828
5.22.3.72	insert	828
5.22.3.73	insert	829
5.22.3.74	insert	829
5.22.3.75	length	830
5.22.3.76	max_size	830
5.22.3.77	operator+=	831
5.22.3.78	operator+=	831
5.22.3.79	operator+=	831
5.22.3.80	operator+=	832
5.22.3.81	operator=	832
5.22.3.82	operator=	832
5.22.3.83	operator=	833
5.22.3.84	operator=	833
5.22.3.85	operator=	833
5.22.3.86	operator[]	834
5.22.3.87	operator[]	834
5.22.3.88	push_back	835
5.22.3.89	rbegin	835
5.22.3.90	rbegin	835
5.22.3.91	rend	835
5.22.3.92	rend	836
5.22.3.93	replace	836
5.22.3.94	replace	837

5.22.3.95	replace	837
5.22.3.96	replace	838
5.22.3.97	replace	839
5.22.3.98	replace	839
5.22.3.99	replace	840
5.22.3.100	replace	841
5.22.3.101	replace	841
5.22.3.102	replace	842
5.22.3.103	replace	843
5.22.3.104	reserve	843
5.22.3.105	resize	844
5.22.3.106	resize	844
5.22.3.107	rfind	845
5.22.3.108	rfind	845
5.22.3.109	rfind	846
5.22.3.110	rfind	846
5.22.3.111	shrink_to_fit	847
5.22.3.112	size	847
5.22.3.113	substr	847
5.22.3.114	swap	848
5.22.4	Member Data Documentation	848
5.22.4.1	npos	848
5.23	__gnu_cxx::_Caster<_ToType> Struct Template Reference	849
5.23.1	Detailed Description	849
5.24	__gnu_cxx::_Char_types<_CharT> Struct Template Reference	849
5.24.1	Detailed Description	850
5.25	__gnu_cxx::_ExtPtr_allocator<_Tp> Class Template Reference	850
5.25.1	Detailed Description	851
5.26	__gnu_cxx::_Invalid_type Struct Reference	852
5.26.1	Detailed Description	852

5.27	<code>__gnu_cxx::Pointer_adapter< _Storage_policy ></code> Class Template Reference	852
5.27.1	Detailed Description	854
5.28	<code>__gnu_cxx::Relative_pointer_impl< _Tp ></code> Class Template Reference	855
5.28.1	Detailed Description	855
5.29	<code>__gnu_cxx::Relative_pointer_impl< const _Tp ></code> Class Template Reference	856
5.29.1	Detailed Description	856
5.30	<code>__gnu_cxx::Std_pointer_impl< _Tp ></code> Class Template Reference	856
5.30.1	Detailed Description	857
5.31	<code>__gnu_cxx::Unqualified_type< _Tp ></code> Struct Template Reference	857
5.31.1	Detailed Description	857
5.32	<code>__gnu_cxx::annotate_base</code> Struct Reference	858
5.32.1	Detailed Description	858
5.33	<code>__gnu_cxx::array_allocator< _Tp, _Array ></code> Class Template Reference	859
5.33.1	Detailed Description	860
5.34	<code>__gnu_cxx::array_allocator_base< _Tp ></code> Class Template Reference	860
5.34.1	Detailed Description	861
5.35	<code>__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 ></code> Class Template Reference	861
5.35.1	Detailed Description	862
5.35.2	Member Typedef Documentation	863
5.35.2.1	<code>argument_type</code>	863
5.35.2.2	<code>result_type</code>	863
5.36	<code>__gnu_cxx::bitmap_allocator< _Tp ></code> Class Template Reference	863
5.36.1	Detailed Description	865
5.36.2	Member Function Documentation	865
5.36.2.1	<code>_M_allocate_single_object</code>	865
5.36.2.2	<code>_M_deallocate_single_object</code>	865
5.37	<code>__gnu_cxx::char_traits< _CharT ></code> Struct Template Reference	866
5.37.1	Detailed Description	867
5.38	<code>__gnu_cxx::character< V, I, S ></code> Struct Template Reference	867

5.38.1 Detailed Description	868
5.39 <code>__gnu_cxx::condition_base</code> Struct Reference	868
5.39.1 Detailed Description	869
5.40 <code>__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 ></code> Struct Template Reference	869
5.40.1 Detailed Description	870
5.41 <code>__gnu_cxx::constant_unary_fun< _Result, _Argument ></code> Struct Tem- plate Reference	870
5.41.1 Detailed Description	871
5.42 <code>__gnu_cxx::constant_void_fun< _Result ></code> Struct Template Reference	871
5.42.1 Detailed Description	871
5.43 <code>__gnu_cxx::debug_allocator< _Alloc ></code> Class Template Reference . .	872
5.43.1 Detailed Description	872
5.44 <code>__gnu_cxx::enc_filebuf< _CharT ></code> Class Template Reference	873
5.44.1 Detailed Description	876
5.44.2 Member Typedef Documentation	876
5.44.2.1 <code>__streambuf_type</code>	876
5.44.2.2 <code>char_type</code>	877
5.44.2.3 <code>int_type</code>	877
5.44.2.4 <code>off_type</code>	877
5.44.2.5 <code>pos_type</code>	877
5.44.2.6 <code>traits_type</code>	878
5.44.3 Member Function Documentation	878
5.44.3.1 <code>_M_create_pback</code>	878
5.44.3.2 <code>_M_destroy_pback</code>	878
5.44.3.3 <code>_M_set_buffer</code>	878
5.44.3.4 <code>close</code>	879
5.44.3.5 <code>eback</code>	879
5.44.3.6 <code>egptr</code>	879
5.44.3.7 <code>epptr</code>	880
5.44.3.8 <code>gbump</code>	880
5.44.3.9 <code>getloc</code>	880

5.44.3.10 gptr	881
5.44.3.11 imbue	881
5.44.3.12 in_avail	882
5.44.3.13 is_open	882
5.44.3.14 open	882
5.44.3.15 open	883
5.44.3.16 overflow	883
5.44.3.17 pbackfail	884
5.44.3.18 pbase	884
5.44.3.19 pbump	884
5.44.3.20 pptr	885
5.44.3.21 pubimbue	885
5.44.3.22 pubseekoff	885
5.44.3.23 pubseekpos	886
5.44.3.24 pubsetbuf	886
5.44.3.25 pubsync	886
5.44.3.26 sbumpc	887
5.44.3.27 seekoff	887
5.44.3.28 seekpos	887
5.44.3.29 setbuf	888
5.44.3.30 setbuf	888
5.44.3.31 setg	889
5.44.3.32 setp	889
5.44.3.33 sgetc	889
5.44.3.34 sgetn	890
5.44.3.35 showmanyc	890
5.44.3.36 snextc	890
5.44.3.37 sputbackc	891
5.44.3.38 sputc	891
5.44.3.39 sputn	892
5.44.3.40 stossc	892

5.44.3.41	sungetc	892
5.44.3.42	sync	893
5.44.3.43	uflow	893
5.44.3.44	underflow	893
5.44.3.45	xsgetn	894
5.44.3.46	xspn	894
5.44.4	Member Data Documentation	895
5.44.4.1	_M_buf	895
5.44.4.2	_M_buf_locale	895
5.44.4.3	_M_buf_size	895
5.44.4.4	_M_ext_buf	895
5.44.4.5	_M_ext_buf_size	896
5.44.4.6	_M_ext_next	896
5.44.4.7	_M_in_beg	896
5.44.4.8	_M_in_cur	896
5.44.4.9	_M_in_end	897
5.44.4.10	_M_mode	897
5.44.4.11	_M_out_beg	897
5.44.4.12	_M_out_cur	897
5.44.4.13	_M_out_end	898
5.44.4.14	_M_pback	898
5.44.4.15	_M_pback_cur_save	898
5.44.4.16	_M_pback_end_save	899
5.44.4.17	_M_pback_init	899
5.44.4.18	_M_reading	899
5.45	__gnu_cxx::encoding_char_traits<_CharT> Struct Template Reference	899
5.45.1	Detailed Description	901
5.46	__gnu_cxx::encoding_state Class Reference	901
5.46.1	Detailed Description	902
5.47	__gnu_cxx::forced_error Struct Reference	902
5.47.1	Detailed Description	903

5.47.2	Member Function Documentation	903
5.47.2.1	what	903
5.48	<code>__gnu_cxx::free_list</code> Class Reference	904
5.48.1	Detailed Description	904
5.48.2	Member Function Documentation	905
5.48.2.1	<code>_M_clear</code>	905
5.48.2.2	<code>_M_get</code>	905
5.48.2.3	<code>_M_insert</code>	905
5.49	<code>__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc></code> Class Template Reference	905
5.49.1	Detailed Description	907
5.50	<code>__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _-</code> <code>Alloc></code> Class Template Reference	908
5.50.1	Detailed Description	909
5.51	<code>__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc></code> Class Template Reference	910
5.51.1	Detailed Description	911
5.52	<code>__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc></code> Class Template Reference	912
5.52.1	Detailed Description	913
5.53	<code>__gnu_cxx::limit_condition</code> Struct Reference	914
5.53.1	Detailed Description	915
5.54	<code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference	915
5.54.1	Detailed Description	915
5.55	<code>__gnu_cxx::limit_condition::limit_adjustor</code> Struct Reference	915
5.55.1	Detailed Description	915
5.56	<code>__gnu_cxx::limit_condition::never_adjustor</code> Struct Reference	916
5.56.1	Detailed Description	916
5.57	<code>__gnu_cxx::malloc_allocator<_Tp></code> Class Template Reference	916
5.57.1	Detailed Description	917
5.58	<code>__gnu_cxx::new_allocator<_Tp></code> Class Template Reference	917
5.58.1	Detailed Description	919

5.59	__gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference	919
5.59.1	Detailed Description	920
5.59.2	Member Typedef Documentation	920
5.59.2.1	first_argument_type	920
5.59.2.2	result_type	920
5.59.2.3	second_argument_type	920
5.60	__gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference	920
5.60.1	Detailed Description	921
5.60.2	Member Typedef Documentation	921
5.60.2.1	first_argument_type	921
5.60.2.2	result_type	921
5.60.2.3	second_argument_type	921
5.61	__gnu_cxx::random_condition Struct Reference	922
5.61.1	Detailed Description	923
5.62	__gnu_cxx::random_condition::always_adjustor Struct Reference	923
5.62.1	Detailed Description	923
5.63	__gnu_cxx::random_condition::group_adjustor Struct Reference	923
5.63.1	Detailed Description	923
5.64	__gnu_cxx::random_condition::never_adjustor Struct Reference	924
5.64.1	Detailed Description	924
5.65	__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference	924
5.65.1	Detailed Description	927
5.66	__gnu_cxx::recursive_init_error Class Reference	927
5.66.1	Detailed Description	928
5.66.2	Member Function Documentation	928
5.66.2.1	what	928
5.67	__gnu_cxx::rope< _CharT, _Alloc > Class Template Reference	929
5.67.1	Detailed Description	935
5.68	__gnu_cxx::select1st< _Pair > Struct Template Reference	935
5.68.1	Detailed Description	936

5.68.2	Member Typedef Documentation	936
5.68.2.1	argument_type	936
5.68.2.2	result_type	936
5.69	<code>__gnu_cxx::select2nd<_Pair></code> Struct Template Reference	936
5.69.1	Detailed Description	937
5.69.2	Member Typedef Documentation	937
5.69.2.1	argument_type	937
5.69.2.2	result_type	937
5.70	<code>__gnu_cxx::slist<_Tp, _Alloc></code> Class Template Reference	937
5.70.1	Detailed Description	940
5.71	<code>__gnu_cxx::stdio_filebuf<_CharT, _Traits></code> Class Template Reference	940
5.71.1	Detailed Description	945
5.71.2	Member Typedef Documentation	945
5.71.2.1	<code>__streambuf_type</code>	945
5.71.2.2	<code>char_type</code>	945
5.71.2.3	<code>int_type</code>	945
5.71.2.4	<code>off_type</code>	946
5.71.2.5	<code>pos_type</code>	946
5.71.2.6	<code>traits_type</code>	946
5.71.3	Constructor & Destructor Documentation	946
5.71.3.1	<code>stdio_filebuf</code>	946
5.71.3.2	<code>stdio_filebuf</code>	947
5.71.3.3	<code>stdio_filebuf</code>	947
5.71.3.4	<code>~stdio_filebuf</code>	948
5.71.4	Member Function Documentation	948
5.71.4.1	<code>_M_create_pback</code>	948
5.71.4.2	<code>_M_destroy_pback</code>	948
5.71.4.3	<code>_M_set_buffer</code>	948
5.71.4.4	<code>close</code>	949
5.71.4.5	<code>eback</code>	949
5.71.4.6	<code>egptr</code>	949

5.71.4.7	epptr	950
5.71.4.8	fd	950
5.71.4.9	file	951
5.71.4.10	gbump	951
5.71.4.11	getloc	951
5.71.4.12	gptr	951
5.71.4.13	imbue	952
5.71.4.14	in_avail	953
5.71.4.15	is_open	953
5.71.4.16	open	953
5.71.4.17	open	954
5.71.4.18	overflow	954
5.71.4.19	pbackfail	955
5.71.4.20	pbase	956
5.71.4.21	pbump	956
5.71.4.22	pptr	956
5.71.4.23	pubimbue	957
5.71.4.24	pubseekoff	957
5.71.4.25	pubseekpos	958
5.71.4.26	pubsetbuf	958
5.71.4.27	pubsync	958
5.71.4.28	sbumpc	959
5.71.4.29	seekoff	959
5.71.4.30	seekpos	959
5.71.4.31	setbuf	960
5.71.4.32	setbuf	960
5.71.4.33	setg	961
5.71.4.34	setp	961
5.71.4.35	sgetc	961
5.71.4.36	sgetn	962
5.71.4.37	showmanyc	962

5.71.4.38	snextc	963
5.71.4.39	sputbackc	963
5.71.4.40	sputc	964
5.71.4.41	sputn	964
5.71.4.42	stossc	964
5.71.4.43	sungetc	965
5.71.4.44	sync	965
5.71.4.45	uflow	966
5.71.4.46	underflow	966
5.71.4.47	xsgetn	967
5.71.4.48	xspn	967
5.71.5	Member Data Documentation	968
5.71.5.1	_M_buf	968
5.71.5.2	_M_buf_locale	968
5.71.5.3	_M_buf_size	968
5.71.5.4	_M_ext_buf	969
5.71.5.5	_M_ext_buf_size	969
5.71.5.6	_M_ext_next	969
5.71.5.7	_M_in_beg	969
5.71.5.8	_M_in_cur	970
5.71.5.9	_M_in_end	970
5.71.5.10	_M_mode	970
5.71.5.11	_M_out_beg	971
5.71.5.12	_M_out_cur	971
5.71.5.13	_M_out_end	971
5.71.5.14	_M_pback	972
5.71.5.15	_M_pback_cur_save	972
5.71.5.16	_M_pback_end_save	972
5.71.5.17	_M_pback_init	973
5.71.5.18	_M_reading	973

5.72 <code>__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits ></code> Class Template Reference	973
5.72.1 Detailed Description	977
5.72.2 Member Typedef Documentation	977
5.72.2.1 <code>__streambuf_type</code>	977
5.72.2.2 <code>char_type</code>	978
5.72.2.3 <code>int_type</code>	978
5.72.2.4 <code>off_type</code>	978
5.72.2.5 <code>pos_type</code>	978
5.72.2.6 <code>traits_type</code>	979
5.72.3 Member Function Documentation	979
5.72.3.1 <code>eback</code>	979
5.72.3.2 <code>egptr</code>	979
5.72.3.3 <code>epptr</code>	980
5.72.3.4 <code>file</code>	980
5.72.3.5 <code>gbump</code>	980
5.72.3.6 <code>getloc</code>	981
5.72.3.7 <code>gptr</code>	981
5.72.3.8 <code>imbue</code>	981
5.72.3.9 <code>in_avail</code>	982
5.72.3.10 <code>overflow</code>	982
5.72.3.11 <code>pbackfail</code>	983
5.72.3.12 <code>pbase</code>	984
5.72.3.13 <code>pbump</code>	984
5.72.3.14 <code>pptr</code>	984
5.72.3.15 <code>pubimbue</code>	985
5.72.3.16 <code>pubseekoff</code>	985
5.72.3.17 <code>pubseekpos</code>	986
5.72.3.18 <code>pubsetbuf</code>	986
5.72.3.19 <code>pubsync</code>	986
5.72.3.20 <code>sbumpc</code>	987

5.72.3.21	seekoff	987
5.72.3.22	seekpos	987
5.72.3.23	setbuf	988
5.72.3.24	setg	988
5.72.3.25	setp	988
5.72.3.26	sgetc	989
5.72.3.27	sgetn	989
5.72.3.28	showmanyc	990
5.72.3.29	snextc	990
5.72.3.30	sputbackc	991
5.72.3.31	putc	991
5.72.3.32	putn	992
5.72.3.33	stossc	992
5.72.3.34	sungetc	992
5.72.3.35	sync	993
5.72.3.36	uflow	993
5.72.3.37	underflow	993
5.72.3.38	xsgetn	994
5.72.3.39	xspn	995
5.72.4	Member Data Documentation	995
5.72.4.1	_M_buf_locale	995
5.72.4.2	_M_in_beg	995
5.72.4.3	_M_in_cur	996
5.72.4.4	_M_in_end	996
5.72.4.5	_M_out_beg	996
5.72.4.6	_M_out_cur	997
5.72.4.7	_M_out_end	997
5.73	__gnu_cxx::subtractive_rng Class Reference	998
5.73.1	Detailed Description	998
5.73.2	Member Typedef Documentation	999
5.73.2.1	argument_type	999

5.73.2.2	result_type	999
5.73.3	Constructor & Destructor Documentation	999
5.73.3.1	subtractive_rng	999
5.73.3.2	subtractive_rng	999
5.73.4	Member Function Documentation	999
5.73.4.1	operator()	999
5.74	__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp > Struct Template Reference	1000
5.74.1	Detailed Description	1001
5.74.2	Constructor & Destructor Documentation	1001
5.74.2.1	temporary_buffer	1001
5.74.2.2	~temporary_buffer	1001
5.74.3	Member Function Documentation	1002
5.74.3.1	begin	1002
5.74.3.2	end	1002
5.74.3.3	requested_size	1002
5.74.3.4	size	1002
5.75	__gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference	1003
5.75.1	Detailed Description	1004
5.76	__gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference	1004
5.76.1	Detailed Description	1005
5.77	__gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference	1006
5.77.1	Detailed Description	1007
5.78	__gnu_cxx::throw_value_base< _Cond > Struct Template Reference	1007
5.78.1	Detailed Description	1008
5.79	__gnu_cxx::throw_value_limit Struct Reference	1008
5.79.1	Detailed Description	1010
5.80	__gnu_cxx::throw_value_random Struct Reference	1010
5.80.1	Detailed Description	1011
5.81	__gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference	1011

5.81.1 Detailed Description	1012
5.81.2 Member Typedef Documentation	1013
5.81.2.1 argument_type	1013
5.81.2.2 result_type	1013
5.82 __gnu_debug::__is_same<_Type1, _Type2 > Struct Template Refer- ence	1013
5.82.1 Detailed Description	1013
5.83 __gnu_debug::__After_nth_from<_Iterator > Class Template Reference	1014
5.83.1 Detailed Description	1014
5.84 __gnu_debug::__Not_equal_to<_Type > Class Template Reference .	1014
5.84.1 Detailed Description	1014
5.85 __gnu_debug::__Safe_iterator<_Iterator, _Sequence > Class Template Reference	1015
5.85.1 Detailed Description	1017
5.85.2 Constructor & Destructor Documentation	1017
5.85.2.1 _Safe_iterator	1017
5.85.2.2 _Safe_iterator	1018
5.85.2.3 _Safe_iterator	1018
5.85.2.4 _Safe_iterator	1018
5.85.3 Member Function Documentation	1019
5.85.3.1 _M_attach	1019
5.85.3.2 _M_attach	1019
5.85.3.3 _M_attach_single	1019
5.85.3.4 _M_attach_single	1019
5.85.3.5 _M_attached_to	1019
5.85.3.6 _M_can_compare	1020
5.85.3.7 _M_dereferenceable	1020
5.85.3.8 _M_detach	1020
5.85.3.9 _M_detach_single	1020
5.85.3.10 _M_get_distance	1020
5.85.3.11 _M_get_mutex	1021
5.85.3.12 _M_incrementable	1021

5.85.3.13	<code>_M_invalidate</code>	1021
5.85.3.14	<code>_M_invalidate_single</code>	1021
5.85.3.15	<code>_M_is_begin</code>	1022
5.85.3.16	<code>_M_is_end</code>	1022
5.85.3.17	<code>_M_singular</code>	1022
5.85.3.18	<code>base</code>	1022
5.85.3.19	<code>operator_iterator</code>	1023
5.85.3.20	<code>operator*</code>	1023
5.85.3.21	<code>operator++</code>	1023
5.85.3.22	<code>operator++</code>	1024
5.85.3.23	<code>operator--</code>	1024
5.85.3.24	<code>operator--</code>	1024
5.85.3.25	<code>operator-></code>	1024
5.85.3.26	<code>operator=</code>	1025
5.85.4	Member Data Documentation	1025
5.85.4.1	<code>_M_next</code>	1025
5.85.4.2	<code>_M_prior</code>	1025
5.85.4.3	<code>_M_sequence</code>	1026
5.85.4.4	<code>_M_version</code>	1026
5.86	<code>__gnu_debug::Safe_iterator_base</code> Class Reference	1026
5.86.1	Detailed Description	1028
5.86.2	Constructor & Destructor Documentation	1028
5.86.2.1	<code>_Safe_iterator_base</code>	1028
5.86.2.2	<code>_Safe_iterator_base</code>	1028
5.86.2.3	<code>_Safe_iterator_base</code>	1028
5.86.3	Member Function Documentation	1029
5.86.3.1	<code>_M_attach</code>	1029
5.86.3.2	<code>_M_attach_single</code>	1029
5.86.3.3	<code>_M_attached_to</code>	1029
5.86.3.4	<code>_M_can_compare</code>	1029
5.86.3.5	<code>_M_detach</code>	1029

5.86.3.6	<code>_M_detach_single</code>	1029
5.86.3.7	<code>_M_get_mutex</code>	1030
5.86.3.8	<code>_M_singular</code>	1030
5.86.4	Member Data Documentation	1030
5.86.4.1	<code>_M_next</code>	1030
5.86.4.2	<code>_M_prior</code>	1030
5.86.4.3	<code>_M_sequence</code>	1030
5.86.4.4	<code>_M_version</code>	1031
5.87	<code>__gnu_debug::_Safe_sequence<_Sequence></code> Class Template Reference	1031
5.87.1	Detailed Description	1032
5.87.2	Member Function Documentation	1033
5.87.2.1	<code>_M_detach_all</code>	1033
5.87.2.2	<code>_M_detach_singular</code>	1033
5.87.2.3	<code>_M_get_mutex</code>	1033
5.87.2.4	<code>_M_invalidate_all</code>	1033
5.87.2.5	<code>_M_invalidate_if</code>	1033
5.87.2.6	<code>_M_revalidate_singular</code>	1034
5.87.2.7	<code>_M_swap</code>	1034
5.87.2.8	<code>_M_transfer_iter</code>	1034
5.87.3	Member Data Documentation	1034
5.87.3.1	<code>_M_const_iterators</code>	1034
5.87.3.2	<code>_M_iterators</code>	1034
5.87.3.3	<code>_M_version</code>	1035
5.88	<code>__gnu_debug::_Safe_sequence_base</code> Class Reference	1035
5.88.1	Detailed Description	1036
5.88.2	Constructor & Destructor Documentation	1036
5.88.2.1	<code>~_Safe_sequence_base</code>	1036
5.88.3	Member Function Documentation	1037
5.88.3.1	<code>_M_detach_all</code>	1037
5.88.3.2	<code>_M_detach_singular</code>	1037

5.88.3.3	_M_get_mutex	1037
5.88.3.4	_M_invalidate_all	1037
5.88.3.5	_M_revalidate_singular	1037
5.88.3.6	_M_swap	1037
5.88.4	Member Data Documentation	1038
5.88.4.1	_M_const_iterators	1038
5.88.4.2	_M_iterators	1038
5.88.4.3	_M_version	1038
5.89	__gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class	
	Template Reference	1038
5.89.1	Detailed Description	1047
5.89.2	Member Function Documentation	1047
5.89.2.1	_M_detach_all	1047
5.89.2.2	_M_detach_singular	1047
5.89.2.3	_M_get_mutex	1047
5.89.2.4	_M_invalidate_all	1047
5.89.2.5	_M_invalidate_if	1048
5.89.2.6	_M_revalidate_singular	1048
5.89.2.7	_M_swap	1048
5.89.2.8	_M_transfer_iter	1048
5.89.2.9	append	1048
5.89.2.10	append	1049
5.89.2.11	append	1049
5.89.2.12	append	1049
5.89.2.13	append	1050
5.89.2.14	append	1050
5.89.2.15	append	1050
5.89.2.16	assign	1051
5.89.2.17	assign	1051
5.89.2.18	assign	1052
5.89.2.19	assign	1052

5.89.2.20 assign	1052
5.89.2.21 assign	1053
5.89.2.22 assign	1053
5.89.2.23 assign	1054
5.89.2.24 at	1054
5.89.2.25 at	1054
5.89.2.26 back	1055
5.89.2.27 back	1055
5.89.2.28 begin	1055
5.89.2.29 begin	1055
5.89.2.30 c_str	1056
5.89.2.31 capacity	1056
5.89.2.32 cbegin	1056
5.89.2.33 cend	1056
5.89.2.34 clear	1056
5.89.2.35 compare	1056
5.89.2.36 compare	1057
5.89.2.37 compare	1057
5.89.2.38 compare	1058
5.89.2.39 compare	1059
5.89.2.40 compare	1059
5.89.2.41 copy	1060
5.89.2.42 crbegin	1060
5.89.2.43 crend	1060
5.89.2.44 data	1061
5.89.2.45 empty	1061
5.89.2.46 end	1061
5.89.2.47 end	1061
5.89.2.48 erase	1061
5.89.2.49 erase	1062
5.89.2.50 erase	1062

5.89.2.51 find	1063
5.89.2.52 find	1063
5.89.2.53 find	1063
5.89.2.54 find	1064
5.89.2.55 find_first_not_of	1064
5.89.2.56 find_first_not_of	1065
5.89.2.57 find_first_not_of	1065
5.89.2.58 find_first_not_of	1065
5.89.2.59 find_first_of	1066
5.89.2.60 find_first_of	1066
5.89.2.61 find_first_of	1067
5.89.2.62 find_first_of	1067
5.89.2.63 find_last_not_of	1068
5.89.2.64 find_last_not_of	1068
5.89.2.65 find_last_not_of	1068
5.89.2.66 find_last_not_of	1069
5.89.2.67 find_last_of	1069
5.89.2.68 find_last_of	1070
5.89.2.69 find_last_of	1070
5.89.2.70 find_last_of	1071
5.89.2.71 front	1071
5.89.2.72 front	1071
5.89.2.73 get_allocator	1071
5.89.2.74 insert	1071
5.89.2.75 insert	1072
5.89.2.76 insert	1073
5.89.2.77 insert	1073
5.89.2.78 insert	1073
5.89.2.79 insert	1074
5.89.2.80 insert	1075
5.89.2.81 insert	1075

5.89.2.82 insert	1076
5.89.2.83 length	1076
5.89.2.84 max_size	1076
5.89.2.85 operator+=	1077
5.89.2.86 operator+=	1077
5.89.2.87 operator+=	1077
5.89.2.88 operator+=	1078
5.89.2.89 operator[]	1078
5.89.2.90 operator[]	1078
5.89.2.91 push_back	1079
5.89.2.92 rbegin	1079
5.89.2.93 rbegin	1079
5.89.2.94 rend	1079
5.89.2.95 rend	1080
5.89.2.96 replace	1080
5.89.2.97 replace	1080
5.89.2.98 replace	1081
5.89.2.99 replace	1081
5.89.2.100replace	1082
5.89.2.101replace	1083
5.89.2.102replace	1083
5.89.2.103replace	1084
5.89.2.104replace	1084
5.89.2.105replace	1085
5.89.2.106replace	1086
5.89.2.107reserve	1086
5.89.2.108resize	1087
5.89.2.109resize	1087
5.89.2.110rfind	1087
5.89.2.111rfind	1088
5.89.2.112rfind	1088

5.89.2.113	find	1088
5.89.2.114	shrink_to_fit	1089
5.89.2.115	size	1089
5.89.2.116	substr	1089
5.89.2.117	swap	1090
5.89.3	Member Data Documentation	1090
5.89.3.1	_M_const_iterators	1090
5.89.3.2	_M_iterators	1090
5.89.3.3	_M_version	1090
5.89.3.4	npos	1091
5.90	__gnu_parallel::__accumulate_binop_reduct<_BinOp> Struct Template Reference	1091
5.90.1	Detailed Description	1091
5.91	__gnu_parallel::__accumulate_selector<_It> Struct Template Reference	1092
5.91.1	Detailed Description	1092
5.91.2	Member Function Documentation	1093
5.91.2.1	operator()	1093
5.91.3	Member Data Documentation	1093
5.91.3.1	_M_finish_iterator	1093
5.92	__gnu_parallel::__adjacent_difference_selector<_It> Struct Template Reference	1093
5.92.1	Detailed Description	1094
5.92.2	Member Data Documentation	1094
5.92.2.1	_M_finish_iterator	1094
5.93	__gnu_parallel::__adjacent_find_selector Struct Reference	1095
5.93.1	Detailed Description	1095
5.93.2	Member Function Documentation	1096
5.93.2.1	_M_sequential_algorithm	1096
5.93.2.2	operator()	1096
5.94	__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType> Class Template Reference	1096

5.94.1	Detailed Description	1097
5.94.2	Member Typedef Documentation	1098
5.94.2.1	argument_type	1098
5.94.2.2	result_type	1098
5.95	<code>__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType ></code> Class Template Reference . . .	1098
5.95.1	Detailed Description	1099
5.95.2	Member Typedef Documentation	1100
5.95.2.1	argument_type	1100
5.95.2.2	result_type	1100
5.96	<code>__gnu_parallel::__count_if_selector< _It, _Diff ></code> Struct Template Reference	1100
5.96.1	Detailed Description	1101
5.96.2	Member Function Documentation	1101
5.96.2.1	operator()	1101
5.96.3	Member Data Documentation	1101
5.96.3.1	_M_finish_iterator	1101
5.97	<code>__gnu_parallel::__count_selector< _It, _Diff ></code> Struct Template Reference	1102
5.97.1	Detailed Description	1103
5.97.2	Member Function Documentation	1103
5.97.2.1	operator()	1103
5.97.3	Member Data Documentation	1103
5.97.3.1	_M_finish_iterator	1103
5.98	<code>__gnu_parallel::__fill_selector< _It ></code> Struct Template Reference . . .	1104
5.98.1	Detailed Description	1104
5.98.2	Member Function Documentation	1105
5.98.2.1	operator()	1105
5.98.3	Member Data Documentation	1105
5.98.3.1	_M_finish_iterator	1105
5.99	<code>__gnu_parallel::__find_first_of_selector< _FIterator ></code> Struct Template Reference	1105

5.99.1 Detailed Description	1106
5.99.2 Member Function Documentation	1107
5.99.2.1 _M_sequential_algorithm	1107
5.99.2.2 operator()	1107
5.100 __gnu_parallel::__find_if_selector Struct Reference	1107
5.100.1 Detailed Description	1108
5.100.2 Member Function Documentation	1108
5.100.2.1 _M_sequential_algorithm	1108
5.100.2.2 operator()	1109
5.101 __gnu_parallel::__for_each_selector< _It > Struct Template Reference	1109
5.101.1 Detailed Description	1110
5.101.2 Member Function Documentation	1110
5.101.2.1 operator()	1110
5.101.3 Member Data Documentation	1111
5.101.3.1 _M_finish_iterator	1111
5.102 __gnu_parallel::__generate_selector< _It > Struct Template Reference	1111
5.102.1 Detailed Description	1112
5.102.2 Member Function Documentation	1112
5.102.2.1 operator()	1112
5.102.3 Member Data Documentation	1112
5.102.3.1 _M_finish_iterator	1112
5.103 __gnu_parallel::__generic_find_selector Struct Reference	1113
5.103.1 Detailed Description	1113
5.104 __gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference	1113
5.104.1 Detailed Description	1114
5.104.2 Member Data Documentation	1115
5.104.2.1 _M_finish_iterator	1115
5.105 __gnu_parallel::__identity_selector< _It > Struct Template Reference	1115
5.105.1 Detailed Description	1116
5.105.2 Member Function Documentation	1116

5.105.2.1 operator()	1116
5.105.3 Member Data Documentation	1116
5.105.3.1 _M_finish_iterator	1116
5.106 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference	1117
5.106.1 Detailed Description	1117
5.106.2 Constructor & Destructor Documentation	1118
5.106.2.1 __inner_product_selector	1118
5.106.3 Member Function Documentation	1118
5.106.3.1 operator()	1118
5.106.4 Member Data Documentation	1118
5.106.4.1 __begin1_iterator	1118
5.106.4.2 __begin2_iterator	1119
5.106.4.3 _M_finish_iterator	1119
5.107 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference	1119
5.107.1 Detailed Description	1120
5.108 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference	1120
5.108.1 Detailed Description	1120
5.109 __gnu_parallel::__mismatch_selector Struct Reference	1121
5.109.1 Detailed Description	1121
5.109.2 Member Function Documentation	1122
5.109.2.1 _M_sequential_algorithm	1122
5.109.2.2 operator()	1122
5.110 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	1122
5.110.1 Detailed Description	1123
5.111 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Tem- plate Reference	1123
5.111.1 Detailed Description	1123

5.112	<code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __-</code> <code>sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	1124
5.112.1	Detailed Description	1124
5.113	<code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true,</code> <code>_RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Tem- plate Reference	1124
5.113.1	Detailed Description	1125
5.114	<code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __-</code> <code>sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _-</code> <code>Compare ></code> Struct Template Reference	1125
5.114.1	Detailed Description	1126
5.115	<code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch<</code> <code>false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare</code> <code>></code> Struct Template Reference	1126
5.115.1	Detailed Description	1126
5.116	<code>__gnu_parallel::__replace_if_selector< _It, _Op, _Tp ></code> Struct Tem- plate Reference	1127
5.116.1	Detailed Description	1127
5.116.2	Constructor & Destructor Documentation	1128
5.116.2.1	<code>__replace_if_selector</code>	1128
5.116.3	Member Function Documentation	1128
5.116.3.1	<code>operator()</code>	1128
5.116.4	Member Data Documentation	1128
5.116.4.1	<code>__new_val</code>	1128
5.116.4.2	<code>_M_finish_iterator</code>	1129
5.117	<code>__gnu_parallel::__replace_selector< _It, _Tp ></code> Struct Template Ref- erence	1129
5.117.1	Detailed Description	1130
5.117.2	Constructor & Destructor Documentation	1130
5.117.2.1	<code>__replace_selector</code>	1130
5.117.3	Member Function Documentation	1130
5.117.3.1	<code>operator()</code>	1130
5.117.4	Member Data Documentation	1130

5.117.4.1	<code>__new_val</code>	1130
5.117.4.2	<code>_M_finish_iterator</code>	1131
5.118	<code>__gnu_parallel::__transform1_selector<_It></code> Struct Template Reference	1131
5.118.1	Detailed Description	1132
5.118.2	Member Function Documentation	1132
5.118.2.1	<code>operator()</code>	1132
5.118.3	Member Data Documentation	1132
5.118.3.1	<code>_M_finish_iterator</code>	1132
5.119	<code>__gnu_parallel::__transform2_selector<_It></code> Struct Template Reference	1133
5.119.1	Detailed Description	1133
5.119.2	Member Function Documentation	1134
5.119.2.1	<code>operator()</code>	1134
5.119.3	Member Data Documentation	1134
5.119.3.1	<code>_M_finish_iterator</code>	1134
5.120	<code>__gnu_parallel::__unary_negate<_Predicate, argument_type></code> Class Template Reference	1134
5.120.1	Detailed Description	1135
5.120.2	Member Typedef Documentation	1136
5.120.2.1	<code>argument_type</code>	1136
5.120.2.2	<code>result_type</code>	1136
5.121	<code>__gnu_parallel::_DRandomShufflingGlobalData<_RAIter></code> Struct Template Reference	1136
5.121.1	Detailed Description	1137
5.121.2	Constructor & Destructor Documentation	1137
5.121.2.1	<code>_DRandomShufflingGlobalData</code>	1137
5.121.3	Member Data Documentation	1137
5.121.3.1	<code>_M_bin_proc</code>	1137
5.121.3.2	<code>_M_dist</code>	1138
5.121.3.3	<code>_M_num_bins</code>	1138
5.121.3.4	<code>_M_num_bits</code>	1138

5.121.3.5	<code>_M_source</code>	1138
5.121.3.6	<code>_M_starts</code>	1139
5.121.3.7	<code>_M_temporaries</code>	1139
5.122	<code>__gnu_parallel::_DRSSorterPU< _RAIter, RandomNumberGenerator ></code> Struct Template Reference	1139
5.122.1	Detailed Description	1140
5.122.2	Member Data Documentation	1140
5.122.2.1	<code>__bins_end</code>	1140
5.122.2.2	<code>_M_bins_begin</code>	1140
5.122.2.3	<code>_M_num_threads</code>	1140
5.122.2.4	<code>_M_sd</code>	1141
5.122.2.5	<code>_M_seed</code>	1141
5.123	<code>__gnu_parallel::_DummyReduct</code> Struct Reference	1141
5.123.1	Detailed Description	1141
5.124	<code>__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare ></code> Class Template Reference	1142
5.124.1	Detailed Description	1143
5.124.2	Member Typedef Documentation	1143
5.124.2.1	<code>first_argument_type</code>	1143
5.124.2.2	<code>result_type</code>	1143
5.124.2.3	<code>second_argument_type</code>	1143
5.125	<code>__gnu_parallel::_EqualTo< _T1, _T2 ></code> Struct Template Reference	1143
5.125.1	Detailed Description	1144
5.125.2	Member Typedef Documentation	1145
5.125.2.1	<code>first_argument_type</code>	1145
5.125.2.2	<code>result_type</code>	1145
5.125.2.3	<code>second_argument_type</code>	1145
5.126	<code>__gnu_parallel::_GuardedIterator< _RAIter, _Compare ></code> Class Template Reference	1145
5.126.1	Detailed Description	1146
5.126.2	Constructor & Destructor Documentation	1146
5.126.2.1	<code>_GuardedIterator</code>	1146

5.126.3 Member Function Documentation	1146
5.126.3.1 operator _RAIter	1146
5.126.3.2 operator*	1147
5.126.3.3 operator++	1147
5.126.4 Friends And Related Function Documentation	1147
5.126.4.1 operator<	1147
5.126.4.2 operator<=	1148
5.127 __gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _- IteratorCategory > Class Template Reference	1148
5.127.1 Detailed Description	1150
5.127.2 Member Typedef Documentation	1150
5.127.2.1 first_type	1150
5.127.2.2 second_type	1150
5.127.3 Member Data Documentation	1150
5.127.3.1 first	1150
5.127.3.2 second	1151
5.128 __gnu_parallel::IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _- IteratorCategory > Class Template Reference	1151
5.128.1 Detailed Description	1152
5.129 __gnu_parallel::Job< _DifferenceTp > Struct Template Reference	1152
5.129.1 Detailed Description	1152
5.129.2 Member Data Documentation	1153
5.129.2.1 _M_first	1153
5.129.2.2 _M_last	1153
5.129.2.3 _M_load	1153
5.130 __gnu_parallel::Less< _T1, _T2 > Struct Template Reference	1153
5.130.1 Detailed Description	1154
5.130.2 Member Typedef Documentation	1155
5.130.2.1 first_argument_type	1155
5.130.2.2 result_type	1155
5.130.2.3 second_argument_type	1155

5.131 <code>__gnu_parallel::_Lexicographic< _T1, _T2, _Compare ></code> Class Template Reference	1155
5.131.1 Detailed Description	1156
5.131.2 Member Typedef Documentation	1157
5.131.2.1 <code>first_argument_type</code>	1157
5.131.2.2 <code>result_type</code>	1157
5.131.2.3 <code>second_argument_type</code>	1157
5.132 <code>__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare ></code> Class Template Reference	1157
5.132.1 Detailed Description	1158
5.132.2 Member Typedef Documentation	1159
5.132.2.1 <code>first_argument_type</code>	1159
5.132.2.2 <code>result_type</code>	1159
5.132.2.3 <code>second_argument_type</code>	1159
5.133 <code>__gnu_parallel::_LoserTree< __stable, _Tp, _Compare ></code> Class Template Reference	1159
5.133.1 Detailed Description	1160
5.133.2 Member Function Documentation	1161
5.133.2.1 <code>__delete_min_insert</code>	1161
5.133.2.2 <code>__get_min_source</code>	1161
5.133.2.3 <code>__insert_start</code>	1161
5.133.3 Member Data Documentation	1162
5.133.3.1 <code>_M_comp</code>	1162
5.133.3.2 <code>_M_first_insert</code>	1162
5.133.3.3 <code>_M_log_k</code>	1162
5.133.3.4 <code>_M_losers</code>	1162
5.134 <code>__gnu_parallel::_LoserTree< false, _Tp, _Compare ></code> Class Template Reference	1163
5.134.1 Detailed Description	1164
5.134.2 Member Function Documentation	1164
5.134.2.1 <code>__delete_min_insert</code>	1164
5.134.2.2 <code>__get_min_source</code>	1164

5.134.2.3 <code>__init_winner</code>	1165
5.134.2.4 <code>__insert_start</code>	1165
5.134.3 Member Data Documentation	1166
5.134.3.1 <code>_M_comp</code>	1166
5.134.3.2 <code>_M_first_insert</code>	1166
5.134.3.3 <code>_M_log_k</code>	1166
5.134.3.4 <code>_M_losers</code>	1166
5.135 <code>__gnu_parallel::_LoserTreeBase< _Tp, _Compare ></code> Class Template Reference	1167
5.135.1 Detailed Description	1168
5.135.2 Constructor & Destructor Documentation	1168
5.135.2.1 <code>_LoserTreeBase</code>	1168
5.135.2.2 <code>~_LoserTreeBase</code>	1168
5.135.3 Member Function Documentation	1169
5.135.3.1 <code>__get_min_source</code>	1169
5.135.3.2 <code>__insert_start</code>	1169
5.135.4 Member Data Documentation	1169
5.135.4.1 <code>_M_comp</code>	1169
5.135.4.2 <code>_M_first_insert</code>	1170
5.135.4.3 <code>_M_log_k</code>	1170
5.135.4.4 <code>_M_losers</code>	1170
5.136 <code>__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser</code> Struct Reference	1170
5.136.1 Detailed Description	1171
5.136.2 Member Data Documentation	1171
5.136.2.1 <code>_M_key</code>	1171
5.136.2.2 <code>_M_source</code>	1171
5.136.2.3 <code>_M_sup</code>	1171
5.137 <code>__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare ></code> Class Template Reference	1172
5.137.1 Detailed Description	1173

5.138__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference	1173
5.138.1 Detailed Description	1174
5.139__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference	1174
5.139.1 Detailed Description	1175
5.140__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference	1175
5.140.1 Detailed Description	1176
5.141__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _- Compare > Class Template Reference	1176
5.141.1 Detailed Description	1177
5.142__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference	1177
5.142.1 Detailed Description	1178
5.143__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference	1179
5.143.1 Detailed Description	1180
5.144__gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference	1180
5.144.1 Detailed Description	1180
5.144.2 Member Data Documentation	1181
5.144.2.1 _M_use_pointer	1181
5.145__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference	1181
5.145.1 Detailed Description	1182
5.146__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference	1183
5.146.1 Detailed Description	1184
5.147__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference	1184
5.147.1 Detailed Description	1185
5.148__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference	1185
5.148.1 Detailed Description	1186

5.148.2 Member Typedef Documentation	1187
5.148.2.1 first_argument_type	1187
5.148.2.2 result_type	1187
5.148.2.3 second_argument_type	1187
5.149 __gnu_parallel::_Nothing Struct Reference	1187
5.149.1 Detailed Description	1187
5.149.2 Member Function Documentation	1188
5.149.2.1 operator()	1188
5.150 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference	1188
5.150.1 Detailed Description	1188
5.150.2 Member Data Documentation	1189
5.150.2.1 _M_begin	1189
5.150.2.2 _M_end	1189
5.151 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference	1189
5.151.1 Detailed Description	1190
5.151.2 Member Typedef Documentation	1191
5.151.2.1 first_argument_type	1191
5.151.2.2 result_type	1191
5.151.2.3 second_argument_type	1191
5.152 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference	1191
5.152.1 Detailed Description	1192
5.152.2 Member Data Documentation	1192
5.152.2.1 _M_num_threads	1192
5.152.2.2 _M_offsets	1192
5.152.2.3 _M_pieces	1192
5.152.2.4 _M_samples	1193
5.152.2.5 _M_source	1193
5.152.2.6 _M_starts	1193
5.152.2.7 _M_temporary	1193

5.153__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference	1194
5.153.1 Detailed Description	1194
5.153.2 Constructor & Destructor Documentation	1194
5.153.2.1 _PseudoSequence	1194
5.153.3 Member Function Documentation	1195
5.153.3.1 begin	1195
5.153.3.2 end	1195
5.154__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference	1195
5.154.1 Detailed Description	1196
5.155__gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Ref- erence	1196
5.155.1 Detailed Description	1197
5.155.2 Member Typedef Documentation	1197
5.155.2.1 _Piece	1197
5.155.3 Constructor & Destructor Documentation	1197
5.155.3.1 _QSBThreadLocal	1197
5.155.4 Member Data Documentation	1198
5.155.4.1 _M_elements_leftover	1198
5.155.4.2 _M_global	1198
5.155.4.3 _M_initial	1198
5.155.4.4 _M_leftover_parts	1198
5.155.4.5 _M_num_threads	1198
5.156__gnu_parallel::_RandomNumber Class Reference	1199
5.156.1 Detailed Description	1199
5.156.2 Constructor & Destructor Documentation	1199
5.156.2.1 _RandomNumber	1199
5.156.2.2 _RandomNumber	1199
5.156.3 Member Function Documentation	1200
5.156.3.1 __genrand_bits	1200
5.156.3.2 operator()	1200

5.156.3.3 operator()	1200
5.157 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class	
Template Reference	1200
5.157.1 Detailed Description	1201
5.157.2 Constructor & Destructor Documentation	1202
5.157.2.1 _RestrictedBoundedConcurrentQueue	1202
5.157.2.2 ~_RestrictedBoundedConcurrentQueue	1202
5.157.3 Member Function Documentation	1202
5.157.3.1 pop_back	1202
5.157.3.2 pop_front	1202
5.157.3.3 push_front	1203
5.158 __gnu_parallel::_SamplingSorter< __stable, _RAIter, _-	
StrictWeakOrdering > Struct Template Reference	1203
5.158.1 Detailed Description	1203
5.159 __gnu_parallel::_SamplingSorter< false, _RAIter, _-	
StrictWeakOrdering > Struct Template Reference	1204
5.159.1 Detailed Description	1204
5.160 __gnu_parallel::_Settings Struct Reference	1204
5.160.1 Detailed Description	1206
5.160.2 Member Function Documentation	1206
5.160.2.1 get	1206
5.160.2.2 set	1206
5.160.3 Member Data Documentation	1206
5.160.3.1 accumulate_minimal_n	1206
5.160.3.2 adjacent_difference_minimal_n	1207
5.160.3.3 cache_line_size	1207
5.160.3.4 count_minimal_n	1207
5.160.3.5 fill_minimal_n	1207
5.160.3.6 find_increasing_factor	1207
5.160.3.7 find_initial_block_size	1207
5.160.3.8 find_maximum_block_size	1207
5.160.3.9 find_scale_factor	1208

5.160.3.10	find_sequential_search_size	1208
5.160.3.11	for_each_minimal_n	1208
5.160.3.12	generate_minimal_n	1208
5.160.3.13	L1_cache_size	1208
5.160.3.14	L2_cache_size	1208
5.160.3.15	max_element_minimal_n	1209
5.160.3.16	merge_minimal_n	1209
5.160.3.17	merge_oversampling	1209
5.160.3.18	min_element_minimal_n	1209
5.160.3.19	multiway_merge_minimal_k	1209
5.160.3.20	multiway_merge_minimal_n	1209
5.160.3.21	multiway_merge_oversampling	1210
5.160.3.22	nth_element_minimal_n	1210
5.160.3.23	partial_sort_minimal_n	1210
5.160.3.24	partial_sum_dilation	1210
5.160.3.25	partial_sum_minimal_n	1210
5.160.3.26	partition_chunk_share	1210
5.160.3.27	partition_chunk_size	1211
5.160.3.28	partition_minimal_n	1211
5.160.3.29	qsb_steals	1211
5.160.3.30	random_shuffle_minimal_n	1211
5.160.3.31	replace_minimal_n	1211
5.160.3.32	search_minimal_n	1211
5.160.3.33	set_difference_minimal_n	1211
5.160.3.34	set_intersection_minimal_n	1212
5.160.3.35	set_symmetric_difference_minimal_n	1212
5.160.3.36	set_union_minimal_n	1212
5.160.3.37	sort_minimal_n	1212
5.160.3.38	sort_mwms_oversampling	1212
5.160.3.39	sort_qs_num_samples_preset	1212
5.160.3.40	sort_qsb_base_case_maximal_n	1213

5.160.3.4	ITLB_size	1213
5.160.3.42	transform_minimal_n	1213
5.160.3.43	unique_copy_minimal_n	1213
5.161	<code>__gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator ></code> Struct Template Reference	1213
5.161.1	Detailed Description	1214
5.162	<code>__gnu_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator ></code> Struct Template Reference	1214
5.162.1	Detailed Description	1214
5.163	<code>__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator ></code> Struct Template Reference	1215
5.163.1	Detailed Description	1215
5.164	<code>__gnu_parallel::balanced_quicksort_tag</code> Struct Reference	1215
5.164.1	Detailed Description	1216
5.164.2	Member Function Documentation	1216
5.164.2.1	<code>__get_num_threads</code>	1216
5.164.2.2	<code>set_num_threads</code>	1217
5.165	<code>__gnu_parallel::balanced_tag</code> Struct Reference	1217
5.165.1	Detailed Description	1217
5.165.2	Member Function Documentation	1218
5.165.2.1	<code>__get_num_threads</code>	1218
5.165.2.2	<code>set_num_threads</code>	1218
5.166	<code>__gnu_parallel::constant_size_blocks_tag</code> Struct Reference	1218
5.166.1	Detailed Description	1219
5.167	<code>__gnu_parallel::default_parallel_tag</code> Struct Reference	1219
5.167.1	Detailed Description	1220
5.167.2	Member Function Documentation	1220
5.167.2.1	<code>__get_num_threads</code>	1220
5.167.2.2	<code>set_num_threads</code>	1221
5.168	<code>__gnu_parallel::equal_split_tag</code> Struct Reference	1221
5.168.1	Detailed Description	1221
5.169	<code>__gnu_parallel::exact_tag</code> Struct Reference	1222

5.169.1 Detailed Description	1222
5.169.2 Member Function Documentation	1223
5.169.2.1 __get_num_threads	1223
5.169.2.2 set_num_threads	1223
5.170 __gnu_parallel::find_tag Struct Reference	1223
5.170.1 Detailed Description	1224
5.171 __gnu_parallel::growing_blocks_tag Struct Reference	1224
5.171.1 Detailed Description	1225
5.172 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	1225
5.172.1 Detailed Description	1226
5.172.2 Member Function Documentation	1226
5.172.2.1 __get_num_threads	1226
5.172.2.2 set_num_threads	1227
5.173 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	1227
5.173.1 Detailed Description	1228
5.173.2 Member Function Documentation	1228
5.173.2.1 __get_num_threads	1228
5.173.2.2 set_num_threads	1228
5.174 __gnu_parallel::multiway_mergesort_tag Struct Reference	1228
5.174.1 Detailed Description	1229
5.174.2 Member Function Documentation	1229
5.174.2.1 __get_num_threads	1229
5.174.2.2 set_num_threads	1230
5.175 __gnu_parallel::omp_loop_static_tag Struct Reference	1230
5.175.1 Detailed Description	1231
5.175.2 Member Function Documentation	1231
5.175.2.1 __get_num_threads	1231
5.175.2.2 set_num_threads	1231
5.176 __gnu_parallel::omp_loop_tag Struct Reference	1231
5.176.1 Detailed Description	1232
5.176.2 Member Function Documentation	1232

5.176.2.1	<code>__get_num_threads</code>	1232
5.176.2.2	<code>set_num_threads</code>	1233
5.177	<code>__gnu_parallel::parallel_tag</code> Struct Reference	1233
5.177.1	Detailed Description	1235
5.177.2	Constructor & Destructor Documentation	1235
5.177.2.1	<code>parallel_tag</code>	1235
5.177.2.2	<code>parallel_tag</code>	1235
5.177.3	Member Function Documentation	1235
5.177.3.1	<code>__get_num_threads</code>	1235
5.177.3.2	<code>set_num_threads</code>	1235
5.178	<code>__gnu_parallel::quicksort_tag</code> Struct Reference	1236
5.178.1	Detailed Description	1236
5.178.2	Member Function Documentation	1237
5.178.2.1	<code>__get_num_threads</code>	1237
5.178.2.2	<code>set_num_threads</code>	1237
5.179	<code>__gnu_parallel::sampling_tag</code> Struct Reference	1237
5.179.1	Detailed Description	1238
5.179.2	Member Function Documentation	1238
5.179.2.1	<code>__get_num_threads</code>	1238
5.179.2.2	<code>set_num_threads</code>	1239
5.180	<code>__gnu_parallel::sequential_tag</code> Struct Reference	1239
5.180.1	Detailed Description	1239
5.181	<code>__gnu_parallel::unbalanced_tag</code> Struct Reference	1239
5.181.1	Detailed Description	1240
5.181.2	Member Function Documentation	1240
5.181.2.1	<code>__get_num_threads</code>	1240
5.181.2.2	<code>set_num_threads</code>	1241
5.182	<code>__gnu_pbds::associative_container_tag</code> Struct Reference	1241
5.182.1	Detailed Description	1241
5.183	<code>__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator ></code> Class Template Reference	1242

5.183.1 Detailed Description	1243
5.184__gnu_pbds::basic_hash_tag Struct Reference	1243
5.184.1 Detailed Description	1244
5.185__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_ Tl, Allocator > Class Template Reference	1244
5.185.1 Detailed Description	1245
5.186__gnu_pbds::basic_tree_tag Struct Reference	1245
5.186.1 Detailed Description	1246
5.187__gnu_pbds::binary_heap_tag Struct Reference	1246
5.187.1 Detailed Description	1246
5.188__gnu_pbds::binomial_heap_tag Struct Reference	1247
5.188.1 Detailed Description	1247
5.189__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1248
5.189.1 Detailed Description	1249
5.190__gnu_pbds::cc_hash_tag Struct Reference	1250
5.190.1 Detailed Description	1250
5.191__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Alloca- tor > Class Template Reference	1251
5.191.1 Detailed Description	1252
5.192__gnu_pbds::container_tag Struct Reference	1252
5.192.1 Detailed Description	1252
5.193__gnu_pbds::container_traits< Cntnr > Struct Template Reference . .	1253
5.193.1 Detailed Description	1253
5.194__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false > Struct Template Reference	1253
5.194.1 Detailed Description	1254
5.195__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true > Struct Template Reference	1254
5.195.1 Detailed Description	1255
5.196__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, false > Struct Template Reference	1255

5.196.1 Detailed Description	1256
5.197 __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true > Struct Template Reference	1256
5.197.1 Detailed Description	1257
5.198 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1257
5.198.1 Detailed Description	1259
5.199 __gnu_pbds::gp_hash_tag Struct Reference	1259
5.199.1 Detailed Description	1260
5.200 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator > Class Template Reference	1260
5.200.1 Detailed Description	1262
5.201 __gnu_pbds::list_update_tag Struct Reference	1262
5.201.1 Detailed Description	1263
5.202 __gnu_pbds::null_mapped_type Struct Reference	1263
5.202.1 Detailed Description	1263
5.203 __gnu_pbds::ov_tree_tag Struct Reference	1264
5.203.1 Detailed Description	1264
5.204 __gnu_pbds::pairing_heap_tag Struct Reference	1265
5.204.1 Detailed Description	1265
5.205 __gnu_pbds::pat_trie_tag Struct Reference	1265
5.205.1 Detailed Description	1266
5.206 __gnu_pbds::priority_queue_tag Struct Reference	1267
5.206.1 Detailed Description	1267
5.207 __gnu_pbds::rb_tree_tag Struct Reference	1267
5.207.1 Detailed Description	1268
5.208 __gnu_pbds::rc_binomial_heap_tag Struct Reference	1269
5.208.1 Detailed Description	1269
5.209 __gnu_pbds::sequence_tag Struct Reference	1269
5.209.1 Detailed Description	1270
5.210 __gnu_pbds::splay_tree_tag Struct Reference	1270

5.210.1 Detailed Description	1271
5.211 __gnu_pbds::string_tag Struct Reference	1271
5.211.1 Detailed Description	1272
5.212 __gnu_pbds::thin_heap_tag Struct Reference	1272
5.212.1 Detailed Description	1273
5.213 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator > Class Template Reference	1273
5.213.1 Detailed Description	1275
5.214 __gnu_pbds::tree_tag Struct Reference	1275
5.214.1 Detailed Description	1276
5.215 __gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator > Class Template Reference	1276
5.215.1 Detailed Description	1277
5.216 __gnu_pbds::trie_tag Struct Reference	1278
5.216.1 Detailed Description	1278
5.217 __gnu_profile::__container_size_info Class Reference	1279
5.217.1 Detailed Description	1280
5.218 __gnu_profile::__container_size_stack_info Class Reference	1280
5.218.1 Detailed Description	1281
5.219 __gnu_profile::__hashfunc_info Class Reference	1281
5.219.1 Detailed Description	1282
5.220 __gnu_profile::__hashfunc_stack_info Class Reference	1283
5.220.1 Detailed Description	1284
5.221 __gnu_profile::__list2vector_info Class Reference	1284
5.221.1 Detailed Description	1285
5.222 __gnu_profile::__map2umap_info Class Reference	1285
5.222.1 Detailed Description	1287
5.223 __gnu_profile::__map2umap_stack_info Class Reference	1287
5.223.1 Detailed Description	1288
5.224 __gnu_profile::__object_info_base Class Reference	1288
5.224.1 Detailed Description	1289
5.225 __gnu_profile::__reentrance_guard Struct Reference	1289

5.225.1 Detailed Description	1289
5.226__gnu_profile::__stack_hash Class Reference	1290
5.226.1 Detailed Description	1290
5.227__gnu_profile::__stack_info_base< __object_info > Class Template Reference	1290
5.227.1 Detailed Description	1290
5.228__gnu_profile::__trace_base< __object_info, __stack_info > Class Template Reference	1291
5.228.1 Detailed Description	1291
5.229__gnu_profile::__trace_container_size Class Reference	1292
5.229.1 Detailed Description	1292
5.230__gnu_profile::__trace_hash_func Class Reference	1292
5.230.1 Detailed Description	1293
5.231__gnu_profile::__trace_hashtable_size Class Reference	1294
5.231.1 Detailed Description	1294
5.232__gnu_profile::__trace_map2umap Class Reference	1294
5.232.1 Detailed Description	1295
5.233__gnu_profile::__trace_vector_size Class Reference	1296
5.233.1 Detailed Description	1296
5.234__gnu_profile::__trace_vector_to_list Class Reference	1296
5.234.1 Detailed Description	1298
5.235__gnu_profile::__vector2list_info Class Reference	1298
5.235.1 Detailed Description	1299
5.236__gnu_profile::__vector2list_stack_info Class Reference	1299
5.236.1 Detailed Description	1301
5.237__gnu_profile::__warning_data Struct Reference	1301
5.237.1 Detailed Description	1301
5.238std::__basic_future< _Res > Class Template Reference	1302
5.238.1 Detailed Description	1303
5.238.2 Member Function Documentation	1303
5.238.2.1 _M_get_result	1303

5.239std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class	
Template Reference	1303
5.239.1 Detailed Description	1305
5.239.2 Member Function Documentation	1305
5.239.2.1 do_out	1305
5.239.2.2 in	1306
5.239.2.3 out	1307
5.239.2.4 unshift	1308
5.240std::__ctype_abstract_base< _CharT > Class Template Reference . .	1308
5.240.1 Detailed Description	1311
5.240.2 Member Typedef Documentation	1311
5.240.2.1 char_type	1311
5.240.3 Member Function Documentation	1312
5.240.3.1 do_is	1312
5.240.3.2 do_is	1312
5.240.3.3 do_narrow	1313
5.240.3.4 do_narrow	1313
5.240.3.5 do_scan_is	1314
5.240.3.6 do_scan_not	1314
5.240.3.7 do_tolower	1315
5.240.3.8 do_tolower	1315
5.240.3.9 do_toupper	1316
5.240.3.10do_toupper	1316
5.240.3.11do_widen	1317
5.240.3.12do_widen	1317
5.240.3.13is	1318
5.240.3.14is	1318
5.240.3.15narrow	1319
5.240.3.16narrow	1319
5.240.3.17scan_is	1320
5.240.3.18scan_not	1320

5.240.3.19 tolower	1321
5.240.3.20 tolower	1321
5.240.3.21 toupper	1322
5.240.3.22 toupper	1322
5.240.3.23 widen	1322
5.240.3.24 widen	1323
5.241 std::__debug::bitset< _Nb > Class Template Reference	1324
5.241.1 Detailed Description	1326
5.241.2 Member Function Documentation	1326
5.241.2.1 _M_detach_all	1326
5.241.2.2 _M_detach_singular	1326
5.241.2.3 _M_get_mutex	1326
5.241.2.4 _M_invalidate_all	1327
5.241.2.5 _M_revalidate_singular	1327
5.241.2.6 _M_swap	1327
5.241.3 Member Data Documentation	1327
5.241.3.1 _M_const_iterators	1327
5.241.3.2 _M_iterators	1327
5.241.3.3 _M_version	1328
5.242 std::__debug::deque< _Tp, _Allocator > Class Template Reference	1328
5.242.1 Detailed Description	1331
5.242.2 Member Function Documentation	1331
5.242.2.1 _M_detach_all	1331
5.242.2.2 _M_detach_singular	1331
5.242.2.3 _M_get_mutex	1332
5.242.2.4 _M_invalidate_all	1332
5.242.2.5 _M_invalidate_if	1332
5.242.2.6 _M_revalidate_singular	1332
5.242.2.7 _M_swap	1332
5.242.2.8 _M_transfer_iter	1332
5.242.3 Member Data Documentation	1333

5.242.3.1	_M_const_iterators	1333
5.242.3.2	_M_iterators	1333
5.242.3.3	_M_version	1333
5.243	std::__debug::list< _Tp, _Allocator > Class Template Reference	1333
5.243.1	Detailed Description	1337
5.243.2	Member Function Documentation	1337
5.243.2.1	_M_detach_all	1337
5.243.2.2	_M_detach_singular	1337
5.243.2.3	_M_get_mutex	1337
5.243.2.4	_M_invalidate_all	1338
5.243.2.5	_M_invalidate_if	1338
5.243.2.6	_M_revalidate_singular	1338
5.243.2.7	_M_swap	1338
5.243.2.8	_M_transfer_iter	1338
5.243.3	Member Data Documentation	1338
5.243.3.1	_M_const_iterators	1338
5.243.3.2	_M_iterators	1339
5.243.3.3	_M_version	1339
5.244	std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1339
5.244.1	Detailed Description	1342
5.244.2	Member Function Documentation	1342
5.244.2.1	_M_detach_all	1342
5.244.2.2	_M_detach_singular	1343
5.244.2.3	_M_get_mutex	1343
5.244.2.4	_M_invalidate_all	1343
5.244.2.5	_M_invalidate_if	1343
5.244.2.6	_M_revalidate_singular	1343
5.244.2.7	_M_swap	1344
5.244.2.8	_M_transfer_iter	1344
5.244.3	Member Data Documentation	1344

5.244.3.1	_M_const_iterators	1344
5.244.3.2	_M_iterators	1344
5.244.3.3	_M_version	1344
5.245	std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class	
	Template Reference	1345
5.245.1	Detailed Description	1347
5.245.2	Member Function Documentation	1348
5.245.2.1	_M_detach_all	1348
5.245.2.2	_M_detach_singular	1348
5.245.2.3	_M_get_mutex	1348
5.245.2.4	_M_invalidate_all	1348
5.245.2.5	_M_invalidate_if	1348
5.245.2.6	_M_revalidate_singular	1349
5.245.2.7	_M_swap	1349
5.245.2.8	_M_transfer_iter	1349
5.245.3	Member Data Documentation	1349
5.245.3.1	_M_const_iterators	1349
5.245.3.2	_M_iterators	1349
5.245.3.3	_M_version	1350
5.246	std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	1350
5.246.1	Detailed Description	1353
5.246.2	Member Function Documentation	1353
5.246.2.1	_M_detach_all	1353
5.246.2.2	_M_detach_singular	1353
5.246.2.3	_M_get_mutex	1353
5.246.2.4	_M_invalidate_all	1353
5.246.2.5	_M_invalidate_if	1354
5.246.2.6	_M_revalidate_singular	1354
5.246.2.7	_M_swap	1354
5.246.2.8	_M_transfer_iter	1354
5.246.3	Member Data Documentation	1354

5.246.3.1	_M_const_iterators	1354
5.246.3.2	_M_iterators	1355
5.246.3.3	_M_version	1355
5.247	std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference	1355
5.247.1	Detailed Description	1358
5.247.2	Member Function Documentation	1358
5.247.2.1	_M_detach_all	1358
5.247.2.2	_M_detach_singular	1359
5.247.2.3	_M_get_mutex	1359
5.247.2.4	_M_invalidate_all	1359
5.247.2.5	_M_invalidate_if	1359
5.247.2.6	_M_revalidate_singular	1359
5.247.2.7	_M_swap	1360
5.247.2.8	_M_transfer_iter	1360
5.247.3	Member Data Documentation	1360
5.247.3.1	_M_const_iterators	1360
5.247.3.2	_M_iterators	1360
5.247.3.3	_M_version	1360
5.248	std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1361
5.248.1	Detailed Description	1363
5.248.2	Member Function Documentation	1363
5.248.2.1	_M_detach_all	1363
5.248.2.2	_M_detach_singular	1363
5.248.2.3	_M_get_mutex	1364
5.248.2.4	_M_invalidate_all	1364
5.248.2.5	_M_invalidate_if	1364
5.248.2.6	_M_revalidate_singular	1364
5.248.2.7	_M_swap	1364
5.248.2.8	_M_transfer_iter	1365
5.248.3	Member Data Documentation	1365

5.248.3.1	_M_const_iterators	1365
5.248.3.2	_M_iterators	1365
5.248.3.3	_M_version	1365
5.249	std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1366
5.249.1	Detailed Description	1368
5.249.2	Member Function Documentation	1368
5.249.2.1	_M_detach_all	1368
5.249.2.2	_M_detach_singular	1368
5.249.2.3	_M_get_mutex	1368
5.249.2.4	_M_invalidate_all	1368
5.249.2.5	_M_invalidate_if	1369
5.249.2.6	_M_revalidate_singular	1369
5.249.2.7	_M_swap	1369
5.249.2.8	_M_transfer_iter	1369
5.249.3	Member Data Documentation	1369
5.249.3.1	_M_const_iterators	1369
5.249.3.2	_M_iterators	1370
5.249.3.3	_M_version	1370
5.250	std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1370
5.250.1	Detailed Description	1372
5.250.2	Member Function Documentation	1372
5.250.2.1	_M_detach_all	1372
5.250.2.2	_M_detach_singular	1372
5.250.2.3	_M_get_mutex	1373
5.250.2.4	_M_invalidate_all	1373
5.250.2.5	_M_invalidate_if	1373
5.250.2.6	_M_revalidate_singular	1373
5.250.2.7	_M_swap	1373
5.250.2.8	_M_transfer_iter	1374
5.250.3	Member Data Documentation	1374

5.250.3.1	_M_const_iterators	1374
5.250.3.2	_M_iterators	1374
5.250.3.3	_M_version	1374
5.251	std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1375
5.251.1	Detailed Description	1377
5.251.2	Member Function Documentation	1377
5.251.2.1	_M_detach_all	1377
5.251.2.2	_M_detach_singular	1377
5.251.2.3	_M_get_mutex	1378
5.251.2.4	_M_invalidate_all	1378
5.251.2.5	_M_invalidate_if	1378
5.251.2.6	_M_revalidate_singular	1378
5.251.2.7	_M_swap	1378
5.251.2.8	_M_transfer_iter	1379
5.251.3	Member Data Documentation	1379
5.251.3.1	_M_const_iterators	1379
5.251.3.2	_M_iterators	1379
5.251.3.3	_M_version	1379
5.252	std::__debug::vector< _Tp, _Allocator > Class Template Reference	1380
5.252.1	Detailed Description	1383
5.252.2	Constructor & Destructor Documentation	1383
5.252.2.1	vector	1383
5.252.3	Member Function Documentation	1383
5.252.3.1	_M_detach_all	1383
5.252.3.2	_M_detach_singular	1383
5.252.3.3	_M_get_mutex	1383
5.252.3.4	_M_invalidate_all	1384
5.252.3.5	_M_invalidate_if	1384
5.252.3.6	_M_revalidate_singular	1384
5.252.3.7	_M_swap	1384

5.252.3.8 <code>_M_transfer_iter</code>	1384
5.252.4 Member Data Documentation	1384
5.252.4.1 <code>_M_const_iterators</code>	1384
5.252.4.2 <code>_M_iterators</code>	1385
5.252.4.3 <code>_M_version</code>	1385
5.253 <code>std::__declval_protector<_Tp></code> Struct Template Reference	1385
5.253.1 Detailed Description	1386
5.254 <code>std::__exception_ptr::exception_ptr</code> Class Reference	1386
5.254.1 Detailed Description	1386
5.255 <code>std::__future_base</code> Struct Reference	1387
5.255.1 Detailed Description	1388
5.256 <code>std::__future_base::Ptr<_Res></code> Struct Template Reference	1388
5.256.1 Detailed Description	1388
5.257 <code>std::__future_base::Result<_Res></code> Struct Template Reference	1389
5.257.1 Detailed Description	1389
5.258 <code>std::__future_base::Result<_Res &></code> Struct Template Reference	1390
5.258.1 Detailed Description	1390
5.259 <code>std::__future_base::Result<void></code> Struct Template Reference	1391
5.259.1 Detailed Description	1391
5.260 <code>std::__future_base::Result_base</code> Struct Reference	1391
5.260.1 Detailed Description	1392
5.261 <code>std::__future_base::State</code> Class Reference	1392
5.261.1 Detailed Description	1393
5.262 <code>std::__is_location_invariant<_Tp></code> Struct Template Reference	1393
5.262.1 Detailed Description	1394
5.263 <code>std::__numeric_limits_base</code> Struct Reference	1394
5.263.1 Detailed Description	1395
5.263.2 Member Data Documentation	1395
5.263.2.1 <code>digits</code>	1395
5.263.2.2 <code>digits10</code>	1395
5.263.2.3 <code>has_denorm</code>	1396

5.263.2.4 has_denorm_loss	1396
5.263.2.5 has_infinity	1396
5.263.2.6 has_quiet_NaN	1396
5.263.2.7 has_signaling_NaN	1396
5.263.2.8 is_bounded	1396
5.263.2.9 is_exact	1396
5.263.2.10 is_iec559	1397
5.263.2.11 is_integer	1397
5.263.2.12 is_modulo	1397
5.263.2.13 is_signed	1397
5.263.2.14 is_specialized	1397
5.263.2.15 max_digits10	1397
5.263.2.16 max_exponent	1398
5.263.2.17 max_exponent10	1398
5.263.2.18 min_exponent	1398
5.263.2.19 min_exponent10	1398
5.263.2.20 radix	1398
5.263.2.21 round_style	1398
5.263.2.22 inyness_before	1399
5.263.2.23 traps	1399
5.264 std::__parallel::__CRandNumber< _MustBeInt > Struct Template Reference	1399
5.264.1 Detailed Description	1399
5.265 std::__profile::bitset< _Nb > Class Template Reference	1399
5.265.1 Detailed Description	1401
5.266 std::__profile::deque< _Tp, _Allocator > Class Template Reference	1401
5.266.1 Detailed Description	1403
5.267 std::__profile::list< _Tp, _Allocator > Class Template Reference	1404
5.267.1 Detailed Description	1406
5.268 std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1406
5.268.1 Detailed Description	1408

5.269	<code>std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference</code>	1409
5.269.1	Detailed Description	1410
5.270	<code>std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference</code>	1411
5.270.1	Detailed Description	1412
5.271	<code>std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference</code>	1413
5.271.1	Detailed Description	1414
5.272	<code>std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference</code>	1415
5.272.1	Detailed Description	1416
5.273	<code>std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference</code>	1416
5.273.1	Detailed Description	1417
5.274	<code>std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference</code>	1418
5.274.1	Detailed Description	1419
5.275	<code>std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference</code>	1419
5.275.1	Detailed Description	1420
5.276	<code>std::_Base_bitset< _Nw > Struct Template Reference</code>	1421
5.276.1	Detailed Description	1422
5.276.2	Member Data Documentation	1422
5.276.2.1	<code>_M_w</code>	1422
5.277	<code>std::_Base_bitset< 0 > Struct Template Reference</code>	1422
5.277.1	Detailed Description	1423
5.278	<code>std::_Base_bitset< 1 > Struct Template Reference</code>	1424
5.278.1	Detailed Description	1425
5.279	<code>std::_Build_index_tuple< _Num > Struct Template Reference</code>	1425
5.279.1	Detailed Description	1425
5.280	<code>std::_Deque_base< _Tp, _Alloc > Class Template Reference</code>	1426
5.280.1	Detailed Description	1427

5.280.2 Member Function Documentation	1427
5.280.2.1 _M_initialize_map	1427
5.281 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference	1428
5.281.1 Detailed Description	1429
5.281.2 Member Function Documentation	1430
5.281.2.1 _M_set_node	1430
5.282 std::_Derives_from_binary_function< _Tp > Struct Template Reference	1430
5.282.1 Detailed Description	1430
5.283 std::_Derives_from_unary_function< _Tp > Struct Template Reference	1431
5.283.1 Detailed Description	1431
5.284 std::_Function_base Class Reference	1431
5.284.1 Detailed Description	1432
5.285 std::_Function_to_function_pointer< _Tp, _IsFunctionType > Struct Template Reference	1432
5.285.1 Detailed Description	1433
5.286 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference	1433
5.286.1 Detailed Description	1434
5.287 std::_Fwd_list_const_iterator< _Tp > Struct Template Reference	1435
5.287.1 Detailed Description	1435
5.288 std::_Fwd_list_iterator< _Tp > Struct Template Reference	1436
5.288.1 Detailed Description	1436
5.289 std::_Fwd_list_node< _Tp > Struct Template Reference	1437
5.289.1 Detailed Description	1438
5.290 std::_Fwd_list_node_base Struct Reference	1438
5.290.1 Detailed Description	1439
5.291 std::_Has_result_type_helper< _Tp > Class Template Reference	1439
5.291.1 Detailed Description	1439
5.292 std::_Index_tuple< _Indexes > Struct Template Reference	1440
5.292.1 Detailed Description	1440
5.293 std::_List_base< _Tp, _Alloc > Class Template Reference	1440
5.293.1 Detailed Description	1441

5.294std::_List_const_iterator< _Tp > Struct Template Reference	1442
5.294.1 Detailed Description	1442
5.295std::_List_iterator< _Tp > Struct Template Reference	1443
5.295.1 Detailed Description	1443
5.296std::_List_node< _Tp > Struct Template Reference	1444
5.296.1 Detailed Description	1445
5.296.2 Member Data Documentation	1445
5.296.2.1 _M_data	1445
5.297std::_List_node_base Struct Reference	1445
5.297.1 Detailed Description	1446
5.298std::_Maybe_get_result_type< _Has_result_type, _Functor > Struct Template Reference	1446
5.298.1 Detailed Description	1447
5.299std::_Maybe_unary_or_binary_function< _Res, _ArgTypes > Struct Template Reference	1447
5.299.1 Detailed Description	1447
5.300std::_Maybe_unary_or_binary_function< _Res, _T1 > Struct Tem- plate Reference	1448
5.300.1 Detailed Description	1448
5.300.2 Member Typedef Documentation	1449
5.300.2.1 argument_type	1449
5.300.2.2 result_type	1449
5.301std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 > Struct Template Reference	1449
5.301.1 Detailed Description	1450
5.301.2 Member Typedef Documentation	1451
5.301.2.1 first_argument_type	1451
5.301.2.2 result_type	1451
5.301.2.3 second_argument_type	1451
5.302std::_Maybe_wrap_member_pointer< _Tp > Struct Template Reference	1451
5.302.1 Detailed Description	1452
5.303std::_Maybe_wrap_member_pointer< _Tp _Class::* > Struct Tem- plate Reference	1452

5.303.1 Detailed Description	1452
5.304std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const > Class Template Reference	1453
5.304.1 Detailed Description	1454
5.305std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile > Class Template Reference	1454
5.305.1 Detailed Description	1454
5.306std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile > Class Template Reference	1455
5.306.1 Detailed Description	1456
5.307std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) > Class Template Reference	1456
5.307.1 Detailed Description	1457
5.308std::_Mu< _Arg, false, false > Class Template Reference	1457
5.308.1 Detailed Description	1457
5.309std::_Mu< _Arg, false, true > Class Template Reference	1458
5.309.1 Detailed Description	1458
5.310std::_Mu< _Arg, true, false > Class Template Reference	1458
5.310.1 Detailed Description	1458
5.311std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference	1459
5.311.1 Detailed Description	1459
5.312std::_Placeholder< _Num > Struct Template Reference	1459
5.312.1 Detailed Description	1459
5.313std::_Reference_wrapper_base< _Tp > Struct Template Reference	1460
5.313.1 Detailed Description	1460
5.314std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference	1461
5.314.1 Detailed Description	1461
5.315std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference	1461
5.315.1 Detailed Description	1462
5.316std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference	1462

5.316.1 Detailed Description	1462
5.317std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference	1463
5.317.1 Detailed Description	1464
5.317.2 Constructor & Destructor Documentation	1464
5.317.2.1 _Temporary_buffer	1464
5.317.3 Member Function Documentation	1464
5.317.3.1 begin	1464
5.317.3.2 end	1464
5.317.3.3 requested_size	1465
5.317.3.4 size	1465
5.318std::_Tuple_impl< _Idx > Struct Template Reference	1465
5.318.1 Detailed Description	1465
5.319std::_Tuple_impl< _Idx, _Head, _Tail...> Struct Template Reference	1466
5.319.1 Detailed Description	1467
5.320std::_Vector_base< _Tp, _Alloc > Struct Template Reference	1467
5.320.1 Detailed Description	1468
5.321std::_Weak_result_type< _Functor > Struct Template Reference . . .	1468
5.321.1 Detailed Description	1468
5.322std::_Weak_result_type_impl< _Functor > Struct Template Reference	1469
5.322.1 Detailed Description	1469
5.323std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)> Struct Tem- plate Reference	1469
5.323.1 Detailed Description	1469
5.324std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)> Struct Tem- plate Reference	1470
5.324.1 Detailed Description	1470
5.325std::_Weak_result_type_impl< _Res(_ArgTypes...)> Struct Template Reference	1470
5.325.1 Detailed Description	1471
5.326std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const > Struct Template Reference	1471
5.326.1 Detailed Description	1471

5.327std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile > Struct Template Reference	1471
5.327.1 Detailed Description	1472
5.328std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile > Struct Template Reference	1472
5.328.1 Detailed Description	1472
5.329std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)> Struct Template Reference	1473
5.329.1 Detailed Description	1473
5.330std::add_lvalue_reference< _Tp > Struct Template Reference	1473
5.330.1 Detailed Description	1473
5.331std::add_rvalue_reference< _Tp > Struct Template Reference	1474
5.331.1 Detailed Description	1474
5.332std::adopt_lock_t Struct Reference	1474
5.332.1 Detailed Description	1474
5.333std::aligned_storage< _Len, _Align > Struct Template Reference	1475
5.333.1 Detailed Description	1475
5.334std::allocator< _Tp > Class Template Reference	1475
5.334.1 Detailed Description	1477
5.335std::allocator< void > Class Template Reference	1477
5.335.1 Detailed Description	1477
5.336std::atomic< _Tp > Struct Template Reference	1478
5.336.1 Detailed Description	1478
5.337std::atomic< _Tp * > Struct Template Reference	1478
5.337.1 Detailed Description	1479
5.338std::atomic< bool > Struct Template Reference	1479
5.338.1 Detailed Description	1480
5.339std::atomic< char > Struct Template Reference	1480
5.339.1 Detailed Description	1481
5.340std::atomic< char16_t > Struct Template Reference	1481
5.340.1 Detailed Description	1482
5.341std::atomic< char32_t > Struct Template Reference	1482

5.341.1 Detailed Description	1483
5.342std::atomic< int > Struct Template Reference	1483
5.342.1 Detailed Description	1483
5.343std::atomic< long > Struct Template Reference	1484
5.343.1 Detailed Description	1484
5.344std::atomic< long long > Struct Template Reference	1484
5.344.1 Detailed Description	1485
5.345std::atomic< short > Struct Template Reference	1485
5.345.1 Detailed Description	1486
5.346std::atomic< signed char > Struct Template Reference	1486
5.346.1 Detailed Description	1487
5.347std::atomic< unsigned char > Struct Template Reference	1487
5.347.1 Detailed Description	1488
5.348std::atomic< unsigned int > Struct Template Reference	1488
5.348.1 Detailed Description	1488
5.349std::atomic< unsigned long > Struct Template Reference	1489
5.349.1 Detailed Description	1489
5.350std::atomic< unsigned long long > Struct Template Reference	1489
5.350.1 Detailed Description	1490
5.351std::atomic< unsigned short > Struct Template Reference	1490
5.351.1 Detailed Description	1491
5.352std::atomic< void * > Struct Template Reference	1491
5.352.1 Detailed Description	1492
5.353std::atomic< wchar_t > Struct Template Reference	1492
5.353.1 Detailed Description	1492
5.354std::auto_ptr< _Tp > Class Template Reference	1493
5.354.1 Detailed Description	1493
5.354.2 Member Typedef Documentation	1494
5.354.2.1 element_type	1494
5.354.3 Constructor & Destructor Documentation	1494
5.354.3.1 auto_ptr	1494

5.354.3.2 auto_ptr	1494
5.354.3.3 auto_ptr	1495
5.354.3.4 ~auto_ptr	1495
5.354.3.5 auto_ptr	1495
5.354.4 Member Function Documentation	1496
5.354.4.1 get	1496
5.354.4.2 operator*	1496
5.354.4.3 operator->	1496
5.354.4.4 operator=	1496
5.354.4.5 operator=	1497
5.354.4.6 release	1497
5.354.4.7 reset	1498
5.355std::auto_ptr_ref< _Tp1 > Struct Template Reference	1498
5.355.1 Detailed Description	1498
5.356std::back_insert_iterator< _Container > Class Template Reference	1499
5.356.1 Detailed Description	1500
5.356.2 Member Typedef Documentation	1500
5.356.2.1 container_type	1500
5.356.2.2 difference_type	1500
5.356.2.3 iterator_category	1500
5.356.2.4 pointer	1501
5.356.2.5 reference	1501
5.356.2.6 value_type	1501
5.356.3 Constructor & Destructor Documentation	1501
5.356.3.1 back_insert_iterator	1501
5.356.4 Member Function Documentation	1501
5.356.4.1 operator*	1501
5.356.4.2 operator++	1502
5.356.4.3 operator++	1502
5.356.4.4 operator=	1502
5.357std::bad_alloc Class Reference	1502

5.357.1 Detailed Description	1503
5.357.2 Member Function Documentation	1503
5.357.2.1 what	1503
5.358std::bad_cast Class Reference	1504
5.358.1 Detailed Description	1504
5.358.2 Member Function Documentation	1504
5.358.2.1 what	1504
5.359std::bad_exception Class Reference	1505
5.359.1 Detailed Description	1505
5.359.2 Member Function Documentation	1505
5.359.2.1 what	1505
5.360std::bad_function_call Class Reference	1506
5.360.1 Detailed Description	1506
5.360.2 Member Function Documentation	1506
5.360.2.1 what	1506
5.361std::bad_typeid Class Reference	1507
5.361.1 Detailed Description	1507
5.361.2 Member Function Documentation	1508
5.361.2.1 what	1508
5.362std::basic_filebuf< _CharT, _Traits > Class Template Reference . . .	1508
5.362.1 Detailed Description	1512
5.362.2 Member Typedef Documentation	1512
5.362.2.1 __streambuf_type	1512
5.362.2.2 char_type	1512
5.362.2.3 int_type	1512
5.362.2.4 off_type	1513
5.362.2.5 pos_type	1513
5.362.2.6 traits_type	1513
5.362.3 Constructor & Destructor Documentation	1514
5.362.3.1 basic_filebuf	1514
5.362.3.2 ~basic_filebuf	1514

5.362.4 Member Function Documentation	1514
5.362.4.1 _M_create_pback	1514
5.362.4.2 _M_destroy_pback	1514
5.362.4.3 _M_set_buffer	1515
5.362.4.4 close	1515
5.362.4.5 eback	1515
5.362.4.6 egptr	1516
5.362.4.7 ep_ptr	1516
5.362.4.8 gbump	1517
5.362.4.9 getloc	1517
5.362.4.10 gp_ptr	1517
5.362.4.11 imbue	1518
5.362.4.12 n_avail	1518
5.362.4.13 is_open	1519
5.362.4.14 open	1519
5.362.4.15 open	1520
5.362.4.16 overflow	1520
5.362.4.17 pbackfail	1521
5.362.4.18 pbase	1521
5.362.4.19 pbump	1522
5.362.4.20 pptr	1522
5.362.4.21 ppubimbue	1522
5.362.4.22 pubseekoff	1523
5.362.4.23 pubseekpos	1523
5.362.4.24 pubsetbuf	1524
5.362.4.25 pubsync	1524
5.362.4.26 bumpc	1524
5.362.4.27 seekoff	1525
5.362.4.28 seekpos	1525
5.362.4.29 setbuf	1525
5.362.4.30 setbuf	1526

5.362.4.31setg	1526
5.362.4.32setp	1527
5.362.4.33getc	1527
5.362.4.34getn	1527
5.362.4.35showmanyc	1528
5.362.4.36nextc	1528
5.362.4.37sputbackc	1529
5.362.4.38putc	1529
5.362.4.39sputn	1530
5.362.4.40tosscc	1530
5.362.4.41sungetc	1530
5.362.4.42sync	1531
5.362.4.43uflow	1531
5.362.4.44underflow	1532
5.362.4.45xsgetn	1532
5.362.4.46xsputn	1533
5.362.5 Member Data Documentation	1534
5.362.5.1 _M_buf	1534
5.362.5.2 _M_buf_locale	1534
5.362.5.3 _M_buf_size	1534
5.362.5.4 _M_ext_buf	1534
5.362.5.5 _M_ext_buf_size	1535
5.362.5.6 _M_ext_next	1535
5.362.5.7 _M_in_beg	1535
5.362.5.8 _M_in_cur	1535
5.362.5.9 _M_in_end	1536
5.362.5.10 _M_mode	1536
5.362.5.11 _M_out_beg	1536
5.362.5.12 _M_out_cur	1537
5.362.5.13 _M_out_end	1537
5.362.5.14 _M_pback	1537

5.362.5.15_M_pback_cur_save	1538
5.362.5.16_M_pback_end_save	1538
5.362.5.17_M_pback_init	1538
5.362.5.18_M_reading	1538
5.363std::basic_fstream< _CharT, _Traits > Class Template Reference . .	1539
5.363.1 Detailed Description	1547
5.363.2 Member Typedef Documentation	1547
5.363.2.1 __ctype_type	1547
5.363.2.2 __ctype_type	1547
5.363.2.3 __num_get_type	1547
5.363.2.4 __num_put_type	1548
5.363.2.5 __num_put_type	1548
5.363.2.6 char_type	1548
5.363.2.7 char_type	1548
5.363.2.8 event_callback	1549
5.363.2.9 fmtflags	1549
5.363.2.10int_type	1550
5.363.2.11int_type	1550
5.363.2.12ostate	1550
5.363.2.13off_type	1551
5.363.2.14off_type	1551
5.363.2.15openmode	1551
5.363.2.16pos_type	1552
5.363.2.17pos_type	1552
5.363.2.18seekdir	1552
5.363.2.19traits_type	1553
5.363.2.20traits_type	1553
5.363.3 Member Enumeration Documentation	1553
5.363.3.1 event	1553
5.363.4 Constructor & Destructor Documentation	1553
5.363.4.1 basic_fstream	1553

5.363.4.2 basic_fstream	1554
5.363.4.3 basic_fstream	1554
5.363.4.4 ~basic_fstream	1554
5.363.5 Member Function Documentation	1555
5.363.5.1 _M_getloc	1555
5.363.5.2 _M_write	1555
5.363.5.3 bad	1555
5.363.5.4 clear	1556
5.363.5.5 close	1556
5.363.5.6 copyfmt	1556
5.363.5.7 eof	1557
5.363.5.8 exceptions	1557
5.363.5.9 exceptions	1558
5.363.5.10fail	1558
5.363.5.11fill	1559
5.363.5.12fill	1559
5.363.5.13flags	1559
5.363.5.14flags	1560
5.363.5.15flush	1560
5.363.5.16gcount	1561
5.363.5.17get	1561
5.363.5.18get	1561
5.363.5.19get	1562
5.363.5.20get	1563
5.363.5.21get	1563
5.363.5.22get	1564
5.363.5.23getline	1564
5.363.5.24getline	1565
5.363.5.25getloc	1565
5.363.5.26good	1566
5.363.5.27ignore	1566

5.363.5.28ignore	1567
5.363.5.29ignore	1567
5.363.5.30imbue	1568
5.363.5.31init	1568
5.363.5.32is_open	1569
5.363.5.33isword	1569
5.363.5.34isnarrow	1569
5.363.5.35open	1570
5.363.5.36open	1570
5.363.5.37operator void *	1571
5.363.5.38operator!	1571
5.363.5.39operator<<	1571
5.363.5.40operator<<	1571
5.363.5.41operator<<	1572
5.363.5.42operator<<	1572
5.363.5.43operator<<	1573
5.363.5.44operator<<	1573
5.363.5.45operator<<	1573
5.363.5.46operator<<	1573
5.363.5.47operator<<	1574
5.363.5.48operator<<	1574
5.363.5.49operator<<	1575
5.363.5.50operator<<	1575
5.363.5.51operator<<	1575
5.363.5.52operator<<	1576
5.363.5.53operator<<	1576
5.363.5.54operator<<	1577
5.363.5.55operator<<	1577
5.363.5.56operator>>	1578
5.363.5.57operator>>	1578
5.363.5.58operator>>	1578

5.363.5.59operator>>	1579
5.363.5.60operator>>	1579
5.363.5.61operator>>	1579
5.363.5.62operator>>	1580
5.363.5.63operator>>	1580
5.363.5.64operator>>	1580
5.363.5.65operator>>	1581
5.363.5.66operator>>	1581
5.363.5.67operator>>	1582
5.363.5.68operator>>	1582
5.363.5.69operator>>	1583
5.363.5.70operator>>	1583
5.363.5.71operator>>	1583
5.363.5.72operator>>	1584
5.363.5.73peek	1584
5.363.5.74precision	1585
5.363.5.75precision	1585
5.363.5.76put	1585
5.363.5.77putback	1586
5.363.5.78pword	1586
5.363.5.79rdbuf	1587
5.363.5.80rdbuf	1587
5.363.5.81rdstate	1588
5.363.5.82read	1588
5.363.5.83readsome	1589
5.363.5.84register_callback	1589
5.363.5.85seekg	1590
5.363.5.86seekg	1590
5.363.5.87seekp	1591
5.363.5.88seekp	1591
5.363.5.89setf	1592

5.363.5.90	setf	1592
5.363.5.91	setstate	1593
5.363.5.92	sync	1593
5.363.5.93	sync_with_stdio	1594
5.363.5.94	tellg	1594
5.363.5.95	tellp	1595
5.363.5.96	tie	1595
5.363.5.97	tie	1596
5.363.5.98	unget	1596
5.363.5.99	unsetf	1596
5.363.5.100	width	1597
5.363.5.101	width	1597
5.363.5.102	width	1598
5.363.5.103	write	1598
5.363.5.104	xalloc	1599
5.363.6	Member Data Documentation	1599
5.363.6.1	_M_gcount	1599
5.363.6.2	adjustfield	1599
5.363.6.3	app	1599
5.363.6.4	ate	1600
5.363.6.5	badbit	1600
5.363.6.6	basefield	1600
5.363.6.7	beg	1600
5.363.6.8	binary	1600
5.363.6.9	boolalpha	1601
5.363.6.10	cur	1601
5.363.6.11	dec	1601
5.363.6.12	end	1601
5.363.6.13	eofbit	1601
5.363.6.14	failbit	1602
5.363.6.15	fixed	1602

5.363.6.16floatfield	1602
5.363.6.17goodbit	1602
5.363.6.18hex	1603
5.363.6.19in	1603
5.363.6.20internal	1603
5.363.6.21left	1603
5.363.6.22oct	1604
5.363.6.23out	1604
5.363.6.24right	1604
5.363.6.25scientific	1604
5.363.6.26showbase	1604
5.363.6.27showpoint	1604
5.363.6.28showpos	1604
5.363.6.29skipws	1605
5.363.6.30trunc	1605
5.363.6.31unitbuf	1605
5.363.6.32uppercase	1605
5.364std::basic_ifstream< _CharT, _Traits > Class Template Reference . .	1605
5.364.1 Detailed Description	1612
5.364.2 Member Typedef Documentation	1612
5.364.2.1 __ctype_type	1612
5.364.2.2 __num_get_type	1612
5.364.2.3 __num_put_type	1613
5.364.2.4 char_type	1613
5.364.2.5 event_callback	1613
5.364.2.6 fmtflags	1613
5.364.2.7 int_type	1614
5.364.2.8 iostate	1615
5.364.2.9 off_type	1615
5.364.2.10openmode	1615
5.364.2.11pos_type	1616

5.364.2.12	seekdir	1616
5.364.2.13	traits_type	1616
5.364.3	Member Enumeration Documentation	1616
5.364.3.1	event	1616
5.364.4	Constructor & Destructor Documentation	1617
5.364.4.1	basic_ifstream	1617
5.364.4.2	basic_ifstream	1617
5.364.4.3	basic_ifstream	1617
5.364.4.4	~basic_ifstream	1618
5.364.5	Member Function Documentation	1618
5.364.5.1	_M_getloc	1618
5.364.5.2	bad	1618
5.364.5.3	clear	1619
5.364.5.4	close	1619
5.364.5.5	copyfmt	1619
5.364.5.6	eof	1620
5.364.5.7	exceptions	1620
5.364.5.8	exceptions	1620
5.364.5.9	fail	1621
5.364.5.10	fill	1621
5.364.5.11	ifill	1622
5.364.5.12	iflags	1622
5.364.5.13	iflags	1623
5.364.5.14	igcount	1623
5.364.5.15	iget	1623
5.364.5.16	iget	1624
5.364.5.17	iget	1624
5.364.5.18	iget	1625
5.364.5.19	iget	1626
5.364.5.20	iget	1626
5.364.5.21	igetline	1627

5.364.5.22	getline	1627
5.364.5.23	getloc	1628
5.364.5.24	good	1628
5.364.5.25	ignore	1629
5.364.5.26	ignore	1629
5.364.5.27	ignore	1630
5.364.5.28	mbue	1630
5.364.5.29	nit	1631
5.364.5.30	is_open	1631
5.364.5.31	liword	1631
5.364.5.32	narrow	1632
5.364.5.33	open	1632
5.364.5.34	open	1633
5.364.5.35	operator void *	1633
5.364.5.36	operator!	1633
5.364.5.37	operator>>	1633
5.364.5.38	operator>>	1634
5.364.5.39	operator>>	1634
5.364.5.40	operator>>	1634
5.364.5.41	operator>>	1635
5.364.5.42	operator>>	1635
5.364.5.43	operator>>	1635
5.364.5.44	operator>>	1636
5.364.5.45	operator>>	1636
5.364.5.46	operator>>	1637
5.364.5.47	operator>>	1637
5.364.5.48	operator>>	1638
5.364.5.49	operator>>	1638
5.364.5.50	operator>>	1639
5.364.5.51	operator>>	1639
5.364.5.52	operator>>	1639

5.364.5.53operator>>	1640
5.364.5.54peek	1640
5.364.5.55precision	1640
5.364.5.56precision	1641
5.364.5.57putback	1641
5.364.5.58pword	1642
5.364.5.59rdbuf	1642
5.364.5.60rdbuf	1643
5.364.5.61rdstate	1643
5.364.5.62read	1643
5.364.5.63readsome	1644
5.364.5.64register_callback	1645
5.364.5.65seekg	1645
5.364.5.66seekg	1646
5.364.5.67setf	1646
5.364.5.68setf	1647
5.364.5.69setstate	1647
5.364.5.70sync	1648
5.364.5.71sync_with_stdio	1648
5.364.5.72tellg	1649
5.364.5.73tie	1649
5.364.5.74tie	1650
5.364.5.75unget	1650
5.364.5.76unsetf	1650
5.364.5.77widen	1651
5.364.5.78width	1651
5.364.5.79width	1652
5.364.5.80xalloc	1652
5.364.6 Member Data Documentation	1652
5.364.6.1 _M_gcount	1652
5.364.6.2 adjustfield	1653

5.364.6.3 app	1653
5.364.6.4 ate	1653
5.364.6.5 badbit	1653
5.364.6.6 basefield	1653
5.364.6.7 beg	1654
5.364.6.8 binary	1654
5.364.6.9 boolalpha	1654
5.364.6.10cur	1654
5.364.6.11dec	1654
5.364.6.12end	1654
5.364.6.13eofbit	1655
5.364.6.14failbit	1655
5.364.6.15fixed	1655
5.364.6.16floatfield	1655
5.364.6.17goodbit	1656
5.364.6.18hex	1656
5.364.6.19in	1656
5.364.6.20internal	1656
5.364.6.21left	1657
5.364.6.22oct	1657
5.364.6.23out	1657
5.364.6.24right	1657
5.364.6.25scientific	1657
5.364.6.26showbase	1657
5.364.6.27showpoint	1658
5.364.6.28showpos	1658
5.364.6.29skipws	1658
5.364.6.30trunc	1658
5.364.6.31unitbuf	1658
5.364.6.32uppercase	1658
5.365std::basic_ios< _CharT, _Traits > Class Template Reference	1659

5.365.1 Detailed Description	1662
5.365.2 Member Typedef Documentation	1663
5.365.2.1 __ctype_type	1663
5.365.2.2 __num_get_type	1663
5.365.2.3 __num_put_type	1663
5.365.2.4 char_type	1663
5.365.2.5 event_callback	1664
5.365.2.6 fmtflags	1664
5.365.2.7 int_type	1665
5.365.2.8 iostate	1665
5.365.2.9 off_type	1666
5.365.2.10 openmode	1666
5.365.2.11 pos_type	1667
5.365.2.12 seekdir	1667
5.365.2.13 traits_type	1667
5.365.3 Member Enumeration Documentation	1668
5.365.3.1 event	1668
5.365.4 Constructor & Destructor Documentation	1668
5.365.4.1 basic_ios	1668
5.365.4.2 ~basic_ios	1668
5.365.4.3 basic_ios	1668
5.365.5 Member Function Documentation	1669
5.365.5.1 _M_getloc	1669
5.365.5.2 bad	1669
5.365.5.3 clear	1669
5.365.5.4 copyfmt	1670
5.365.5.5 eof	1670
5.365.5.6 exceptions	1670
5.365.5.7 exceptions	1671
5.365.5.8 fail	1671
5.365.5.9 fill	1672

5.365.5.10fill	1672
5.365.5.11flags	1673
5.365.5.12flags	1673
5.365.5.13getloc	1673
5.365.5.14good	1674
5.365.5.15mbue	1674
5.365.5.16init	1674
5.365.5.17word	1675
5.365.5.18narrow	1675
5.365.5.19operator void *	1676
5.365.5.20operator!	1676
5.365.5.21precision	1676
5.365.5.22precision	1676
5.365.5.23pword	1677
5.365.5.24rdbuf	1677
5.365.5.25rdbuf	1678
5.365.5.26rdstate	1678
5.365.5.27register_callback	1679
5.365.5.28setf	1679
5.365.5.29setf	1679
5.365.5.30setstate	1680
5.365.5.31sync_with_stdio	1680
5.365.5.32tie	1681
5.365.5.33tie	1681
5.365.5.34unsetf	1681
5.365.5.35widen	1682
5.365.5.36width	1682
5.365.5.37width	1683
5.365.5.38xalloc	1683
5.365.6 Member Data Documentation	1683
5.365.6.1 adjustfield	1683

5.365.6.2	app	1683
5.365.6.3	ate	1684
5.365.6.4	badbit	1684
5.365.6.5	basefield	1684
5.365.6.6	beg	1684
5.365.6.7	binary	1684
5.365.6.8	boolalpha	1685
5.365.6.9	cur	1685
5.365.6.10	dec	1685
5.365.6.11	end	1685
5.365.6.12	eofbit	1685
5.365.6.13	failbit	1686
5.365.6.14	fixed	1686
5.365.6.15	floatfield	1686
5.365.6.16	goodbit	1686
5.365.6.17	hex	1687
5.365.6.18	in	1687
5.365.6.19	internal	1687
5.365.6.20	left	1687
5.365.6.21	loct	1688
5.365.6.22	out	1688
5.365.6.23	right	1688
5.365.6.24	scientific	1688
5.365.6.25	showbase	1688
5.365.6.26	showpoint	1688
5.365.6.27	showpos	1688
5.365.6.28	skipws	1689
5.365.6.29	trunc	1689
5.365.6.30	unitbuf	1689
5.365.6.31	uppercase	1689
5.366	std::basic_iostream<_CharT, _Traits> Class Template Reference	1689

5.366.1 Detailed Description	1697
5.366.2 Member Typedef Documentation	1697
5.366.2.1 __ctype_type	1697
5.366.2.2 __ctype_type	1697
5.366.2.3 __num_get_type	1697
5.366.2.4 __num_put_type	1698
5.366.2.5 __num_put_type	1698
5.366.2.6 char_type	1698
5.366.2.7 char_type	1698
5.366.2.8 event_callback	1699
5.366.2.9 fmtflags	1699
5.366.2.10 int_type	1700
5.366.2.11 int_type	1700
5.366.2.12 iostate	1700
5.366.2.13 off_type	1701
5.366.2.14 off_type	1701
5.366.2.15 openmode	1701
5.366.2.16 pos_type	1702
5.366.2.17 pos_type	1702
5.366.2.18 seekdir	1702
5.366.2.19 traits_type	1703
5.366.2.20 traits_type	1703
5.366.3 Member Enumeration Documentation	1703
5.366.3.1 event	1703
5.366.4 Constructor & Destructor Documentation	1704
5.366.4.1 basic_iostream	1704
5.366.4.2 ~basic_iostream	1704
5.366.5 Member Function Documentation	1704
5.366.5.1 _M_getloc	1704
5.366.5.2 _M_write	1705
5.366.5.3 bad	1705

5.366.5.4 clear	1705
5.366.5.5 copyfmt	1706
5.366.5.6 eof	1706
5.366.5.7 exceptions	1707
5.366.5.8 exceptions	1707
5.366.5.9 fail	1708
5.366.5.10fill	1708
5.366.5.11fill	1708
5.366.5.12flags	1709
5.366.5.13flags	1709
5.366.5.14flush	1709
5.366.5.15gcount	1710
5.366.5.16get	1710
5.366.5.17get	1711
5.366.5.18get	1711
5.366.5.19get	1712
5.366.5.20get	1712
5.366.5.21get	1713
5.366.5.22getline	1714
5.366.5.23getline	1714
5.366.5.24getloc	1715
5.366.5.25good	1715
5.366.5.26ignore	1716
5.366.5.27ignore	1716
5.366.5.28ignore	1717
5.366.5.29imbue	1717
5.366.5.30init	1718
5.366.5.31iword	1718
5.366.5.32narrow	1718
5.366.5.33operator void *	1719
5.366.5.34operator!	1719

5.366.5.35operator<<	1719
5.366.5.36operator<<	1720
5.366.5.37operator<<	1720
5.366.5.38operator<<	1721
5.366.5.39operator<<	1721
5.366.5.40operator<<	1721
5.366.5.41operator<<	1721
5.366.5.42operator<<	1722
5.366.5.43operator<<	1722
5.366.5.44operator<<	1723
5.366.5.45operator<<	1723
5.366.5.46operator<<	1723
5.366.5.47operator<<	1724
5.366.5.48operator<<	1724
5.366.5.49operator<<	1725
5.366.5.50operator<<	1725
5.366.5.51operator<<	1726
5.366.5.52operator>>	1726
5.366.5.53operator>>	1726
5.366.5.54operator>>	1727
5.366.5.55operator>>	1727
5.366.5.56operator>>	1728
5.366.5.57operator>>	1728
5.366.5.58operator>>	1728
5.366.5.59operator>>	1729
5.366.5.60operator>>	1729
5.366.5.61operator>>	1730
5.366.5.62operator>>	1730
5.366.5.63operator>>	1730
5.366.5.64operator>>	1731
5.366.5.65operator>>	1731

5.366.5.66operator>>	1731
5.366.5.67operator>>	1732
5.366.5.68operator>>	1732
5.366.5.69peek	1733
5.366.5.70precision	1733
5.366.5.71precision	1733
5.366.5.72put	1734
5.366.5.73putback	1734
5.366.5.74pword	1735
5.366.5.75rdbuf	1735
5.366.5.76rdbuf	1736
5.366.5.77rdstate	1737
5.366.5.78read	1737
5.366.5.79readsome	1738
5.366.5.80register_callback	1738
5.366.5.81seekg	1739
5.366.5.82seekg	1739
5.366.5.83seekp	1740
5.366.5.84seekp	1740
5.366.5.85setf	1741
5.366.5.86setf	1741
5.366.5.87setstate	1742
5.366.5.88sync	1742
5.366.5.89sync_with_stdio	1743
5.366.5.90tellg	1743
5.366.5.91tellp	1744
5.366.5.92tie	1744
5.366.5.93tie	1744
5.366.5.94unget	1745
5.366.5.95unsetf	1745
5.366.5.96widen	1746

5.366.5.97width	1746
5.366.5.98width	1746
5.366.5.99write	1747
5.366.5.100alloc	1747
5.366.6 Member Data Documentation	1748
5.366.6.1 _M_gcount	1748
5.366.6.2 adjustfield	1748
5.366.6.3 app	1748
5.366.6.4 ate	1748
5.366.6.5 badbit	1748
5.366.6.6 basefield	1749
5.366.6.7 beg	1749
5.366.6.8 binary	1749
5.366.6.9 boolalpha	1749
5.366.6.10cur	1750
5.366.6.11dec	1750
5.366.6.12end	1750
5.366.6.13eofbit	1750
5.366.6.14failbit	1750
5.366.6.15fixed	1751
5.366.6.16floatfield	1751
5.366.6.17goodbit	1751
5.366.6.18hex	1751
5.366.6.19in	1752
5.366.6.20internal	1752
5.366.6.21left	1752
5.366.6.22oct	1752
5.366.6.23out	1752
5.366.6.24right	1753
5.366.6.25scientific	1753
5.366.6.26showbase	1753

5.366.6.27	showpoint	1753
5.366.6.28	showpos	1753
5.366.6.29	skipws	1753
5.366.6.30	trunc	1753
5.366.6.31	unitbuf	1754
5.366.6.32	uppercase	1754
5.367	std::basic_istream< _CharT, _Traits > Class Template Reference . .	1754
5.367.1	Detailed Description	1760
5.367.2	Member Typedef Documentation	1760
5.367.2.1	__ctype_type	1760
5.367.2.2	__num_get_type	1761
5.367.2.3	__num_put_type	1761
5.367.2.4	char_type	1761
5.367.2.5	event_callback	1761
5.367.2.6	fmtflags	1762
5.367.2.7	int_type	1763
5.367.2.8	iostate	1763
5.367.2.9	off_type	1763
5.367.2.10	openmode	1763
5.367.2.11	lpos_type	1764
5.367.2.12	seekdir	1764
5.367.2.13	traits_type	1765
5.367.3	Member Enumeration Documentation	1765
5.367.3.1	event	1765
5.367.4	Constructor & Destructor Documentation	1765
5.367.4.1	basic_istream	1765
5.367.4.2	~basic_istream	1765
5.367.5	Member Function Documentation	1766
5.367.5.1	_M_getloc	1766
5.367.5.2	bad	1766
5.367.5.3	clear	1766

5.367.5.4 copyfmt	1767
5.367.5.5 eof	1767
5.367.5.6 exceptions	1768
5.367.5.7 exceptions	1768
5.367.5.8 fail	1769
5.367.5.9 fill	1769
5.367.5.10fill	1769
5.367.5.11flags	1770
5.367.5.12flags	1770
5.367.5.13gcount	1770
5.367.5.14get	1771
5.367.5.15get	1771
5.367.5.16get	1772
5.367.5.17get	1772
5.367.5.18get	1773
5.367.5.19get	1774
5.367.5.20getline	1774
5.367.5.21getline	1775
5.367.5.22getloc	1775
5.367.5.23good	1776
5.367.5.24ignore	1776
5.367.5.25ignore	1777
5.367.5.26ignore	1777
5.367.5.27imbue	1777
5.367.5.28init	1778
5.367.5.29isword	1778
5.367.5.30narrow	1779
5.367.5.31operator void *	1779
5.367.5.32operator!	1779
5.367.5.33operator>>	1780
5.367.5.34operator>>	1780

5.367.5.35operator>>	1781
5.367.5.36operator>>	1781
5.367.5.37operator>>	1781
5.367.5.38operator>>	1782
5.367.5.39operator>>	1782
5.367.5.40operator>>	1782
5.367.5.41operator>>	1783
5.367.5.42operator>>	1783
5.367.5.43operator>>	1783
5.367.5.44operator>>	1784
5.367.5.45operator>>	1784
5.367.5.46operator>>	1784
5.367.5.47operator>>	1785
5.367.5.48operator>>	1785
5.367.5.49operator>>	1786
5.367.5.50peek	1786
5.367.5.51precision	1786
5.367.5.52precision	1787
5.367.5.53putback	1787
5.367.5.54pword	1788
5.367.5.55rdbuf	1788
5.367.5.56rdbuf	1789
5.367.5.57rdstate	1790
5.367.5.58read	1790
5.367.5.59readsome	1791
5.367.5.60register_callback	1791
5.367.5.61seekg	1792
5.367.5.62seekg	1792
5.367.5.63setf	1793
5.367.5.64setf	1793
5.367.5.65setstate	1794

5.367.5.66	sync	1794
5.367.5.67	sync_with_stdio	1795
5.367.5.68	tellg	1795
5.367.5.69	tie	1796
5.367.5.70	tie	1796
5.367.5.71	unget	1796
5.367.5.72	unsetf	1797
5.367.5.73	widen	1797
5.367.5.74	width	1798
5.367.5.75	width	1798
5.367.5.76	xalloc	1798
5.367.6	Member Data Documentation	1799
5.367.6.1	_M_gcount	1799
5.367.6.2	adjustfield	1799
5.367.6.3	app	1799
5.367.6.4	ate	1799
5.367.6.5	badbit	1800
5.367.6.6	basefield	1800
5.367.6.7	beg	1800
5.367.6.8	binary	1800
5.367.6.9	boolalpha	1801
5.367.6.10	cur	1801
5.367.6.11	dec	1801
5.367.6.12	end	1801
5.367.6.13	eofbit	1801
5.367.6.14	failbit	1802
5.367.6.15	fixed	1802
5.367.6.16	floatfield	1802
5.367.6.17	goodbit	1802
5.367.6.18	hex	1803
5.367.6.19	in	1803

5.367.6.20internal	1803
5.367.6.21left	1803
5.367.6.22oct	1803
5.367.6.23out	1803
5.367.6.24right	1804
5.367.6.25scientific	1804
5.367.6.26showbase	1804
5.367.6.27showpoint	1804
5.367.6.28showpos	1804
5.367.6.29skipws	1804
5.367.6.30trunc	1805
5.367.6.31unitbuf	1805
5.367.6.32uppercase	1805
5.368std::basic_istream< _CharT, _Traits >::sentry Class Reference	1805
5.368.1 Detailed Description	1806
5.368.2 Member Typedef Documentation	1806
5.368.2.1 traits_type	1806
5.368.3 Constructor & Destructor Documentation	1806
5.368.3.1 sentry	1806
5.368.4 Member Function Documentation	1807
5.368.4.1 operator bool	1807
5.369std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference	1807
5.369.1 Detailed Description	1814
5.369.2 Member Typedef Documentation	1814
5.369.2.1 __ctype_type	1814
5.369.2.2 __num_get_type	1814
5.369.2.3 __num_put_type	1814
5.369.2.4 char_type	1815
5.369.2.5 event_callback	1815
5.369.2.6 fmtflags	1815

5.369.2.7 int_type	1816
5.369.2.8 iostate	1816
5.369.2.9 off_type	1817
5.369.2.10 openmode	1817
5.369.2.11 pos_type	1818
5.369.2.12 seekdir	1818
5.369.2.13 traits_type	1818
5.369.3 Member Enumeration Documentation	1818
5.369.3.1 event	1818
5.369.4 Constructor & Destructor Documentation	1819
5.369.4.1 basic_istream	1819
5.369.4.2 basic_istream	1819
5.369.4.3 ~basic_istream	1820
5.369.5 Member Function Documentation	1820
5.369.5.1 _M_getloc	1820
5.369.5.2 bad	1820
5.369.5.3 clear	1821
5.369.5.4 copyfmt	1821
5.369.5.5 eof	1821
5.369.5.6 exceptions	1822
5.369.5.7 exceptions	1822
5.369.5.8 fail	1823
5.369.5.9 fill	1823
5.369.5.10 fill	1824
5.369.5.11 flags	1824
5.369.5.12 flags	1824
5.369.5.13 gcount	1825
5.369.5.14 get	1825
5.369.5.15 get	1825
5.369.5.16 get	1826
5.369.5.17 get	1826

5.369.5.18get	1827
5.369.5.19get	1828
5.369.5.20getline	1828
5.369.5.21getline	1829
5.369.5.22getloc	1830
5.369.5.23good	1830
5.369.5.24ignore	1830
5.369.5.25ignore	1831
5.369.5.26ignore	1832
5.369.5.27mbue	1832
5.369.5.28nit	1833
5.369.5.29word	1833
5.369.5.30narrow	1833
5.369.5.31operator void *	1834
5.369.5.32operator!	1834
5.369.5.33operator>>	1834
5.369.5.34operator>>	1835
5.369.5.35operator>>	1835
5.369.5.36operator>>	1835
5.369.5.37operator>>	1836
5.369.5.38operator>>	1836
5.369.5.39operator>>	1837
5.369.5.40operator>>	1837
5.369.5.41operator>>	1838
5.369.5.42operator>>	1838
5.369.5.43operator>>	1838
5.369.5.44operator>>	1839
5.369.5.45operator>>	1839
5.369.5.46operator>>	1839
5.369.5.47operator>>	1840
5.369.5.48operator>>	1840

5.369.5.49operator>>	1841
5.369.5.50peek	1841
5.369.5.51precision	1841
5.369.5.52precision	1842
5.369.5.53putback	1842
5.369.5.54pword	1843
5.369.5.55rdbuf	1843
5.369.5.56rdbuf	1843
5.369.5.57rdstate	1844
5.369.5.58read	1844
5.369.5.59readsome	1845
5.369.5.60register_callback	1846
5.369.5.61seekg	1846
5.369.5.62seekg	1847
5.369.5.63setf	1847
5.369.5.64setf	1848
5.369.5.65setstate	1848
5.369.5.66str	1849
5.369.5.67str	1849
5.369.5.68sync	1849
5.369.5.69sync_with_stdio	1850
5.369.5.70tellg	1850
5.369.5.71tie	1851
5.369.5.72tie	1851
5.369.5.73unset	1851
5.369.5.74unsetf	1852
5.369.5.75widen	1852
5.369.5.76width	1853
5.369.5.77width	1853
5.369.5.78xalloc	1853
5.369.6 Member Data Documentation	1854

5.369.6.1 _M_gcount	1854
5.369.6.2 adjustfield	1854
5.369.6.3 app	1854
5.369.6.4 ate	1854
5.369.6.5 badbit	1855
5.369.6.6 basefield	1855
5.369.6.7 beg	1855
5.369.6.8 binary	1855
5.369.6.9 boolalpha	1856
5.369.6.10cur	1856
5.369.6.11dec	1856
5.369.6.12end	1856
5.369.6.13eofbit	1856
5.369.6.14failbit	1857
5.369.6.15fixed	1857
5.369.6.16floatfield	1857
5.369.6.17goodbit	1857
5.369.6.18hex	1858
5.369.6.19in	1858
5.369.6.20internal	1858
5.369.6.21left	1858
5.369.6.22oct	1858
5.369.6.23out	1859
5.369.6.24right	1859
5.369.6.25scientific	1859
5.369.6.26showbase	1859
5.369.6.27showpoint	1859
5.369.6.28showpos	1859
5.369.6.29skipws	1860
5.369.6.30trunc	1860
5.369.6.31unitbuf	1860

5.369.6.32uppercase	1860
5.370std::basic_ofstream< _CharT, _Traits > Class Template Reference . .	1860
5.370.1 Detailed Description	1866
5.370.2 Member Typedef Documentation	1867
5.370.2.1 __ctype_type	1867
5.370.2.2 __num_get_type	1867
5.370.2.3 __num_put_type	1867
5.370.2.4 char_type	1867
5.370.2.5 event_callback	1868
5.370.2.6 fmtflags	1868
5.370.2.7 int_type	1869
5.370.2.8 iostate	1869
5.370.2.9 off_type	1869
5.370.2.10openmode	1870
5.370.2.11pos_type	1870
5.370.2.12seekdir	1870
5.370.2.13traits_type	1871
5.370.3 Member Enumeration Documentation	1871
5.370.3.1 event	1871
5.370.4 Constructor & Destructor Documentation	1871
5.370.4.1 basic_ofstream	1871
5.370.4.2 basic_ofstream	1871
5.370.4.3 basic_ofstream	1872
5.370.4.4 ~basic_ofstream	1872
5.370.5 Member Function Documentation	1872
5.370.5.1 _M_getloc	1872
5.370.5.2 _M_write	1873
5.370.5.3 bad	1873
5.370.5.4 clear	1874
5.370.5.5 close	1874
5.370.5.6 copyfmt	1874

5.370.5.7 eof	1875
5.370.5.8 exceptions	1875
5.370.5.9 exceptions	1875
5.370.5.10fail	1876
5.370.5.11fill	1876
5.370.5.12fill	1877
5.370.5.13flags	1877
5.370.5.14flags	1877
5.370.5.15flush	1878
5.370.5.16getloc	1878
5.370.5.17good	1879
5.370.5.18mbue	1879
5.370.5.19nit	1879
5.370.5.20s_open	1880
5.370.5.21word	1880
5.370.5.22narrow	1880
5.370.5.23open	1881
5.370.5.24open	1881
5.370.5.25operator void *	1882
5.370.5.26operator!	1882
5.370.5.27operator<<	1882
5.370.5.28operator<<	1882
5.370.5.29operator<<	1883
5.370.5.30operator<<	1883
5.370.5.31operator<<	1884
5.370.5.32operator<<	1884
5.370.5.33operator<<	1884
5.370.5.34operator<<	1885
5.370.5.35operator<<	1885
5.370.5.36operator<<	1886
5.370.5.37operator<<	1886

5.370.5.38operator<<	1886
5.370.5.39operator<<	1887
5.370.5.40operator<<	1887
5.370.5.41operator<<	1888
5.370.5.42operator<<	1888
5.370.5.43operator<<	1888
5.370.5.44precision	1889
5.370.5.45precision	1889
5.370.5.46put	1889
5.370.5.47pword	1890
5.370.5.48rdbuf	1890
5.370.5.49rdbuf	1891
5.370.5.50rdstate	1891
5.370.5.51register_callback	1891
5.370.5.52seekp	1892
5.370.5.53seekp	1892
5.370.5.54setf	1893
5.370.5.55setf	1893
5.370.5.56setstate	1894
5.370.5.57sync_with_stdio	1894
5.370.5.58tellp	1895
5.370.5.59tie	1895
5.370.5.60tie	1895
5.370.5.61unsetf	1896
5.370.5.62widen	1896
5.370.5.63width	1897
5.370.5.64width	1897
5.370.5.65write	1897
5.370.5.66xalloc	1898
5.370.6 Member Data Documentation	1898
5.370.6.1 adjustfield	1898

5.370.6.2 app	1898
5.370.6.3 ate	1899
5.370.6.4 badbit	1899
5.370.6.5 basefield	1899
5.370.6.6 beg	1899
5.370.6.7 binary	1899
5.370.6.8 boolalpha	1900
5.370.6.9 cur	1900
5.370.6.10dec	1900
5.370.6.11end	1900
5.370.6.12eofbit	1900
5.370.6.13failbit	1901
5.370.6.14fixed	1901
5.370.6.15floatfield	1901
5.370.6.16goodbit	1901
5.370.6.17hex	1902
5.370.6.18in	1902
5.370.6.19internal	1902
5.370.6.20left	1902
5.370.6.21loct	1903
5.370.6.22out	1903
5.370.6.23right	1903
5.370.6.24scientific	1903
5.370.6.25showbase	1903
5.370.6.26showpoint	1903
5.370.6.27showpos	1903
5.370.6.28skipws	1904
5.370.6.29trunc	1904
5.370.6.30unitbuf	1904
5.370.6.31uppercase	1904
5.371std::basic_ostream< _CharT, _Traits > Class Template Reference . .	1904

5.371.1 Detailed Description	1910
5.371.2 Member Typedef Documentation	1910
5.371.2.1 __ctype_type	1910
5.371.2.2 __num_get_type	1910
5.371.2.3 __num_put_type	1910
5.371.2.4 char_type	1911
5.371.2.5 event_callback	1911
5.371.2.6 fmtflags	1911
5.371.2.7 int_type	1912
5.371.2.8 iostate	1913
5.371.2.9 off_type	1913
5.371.2.10 openmode	1913
5.371.2.11 pos_type	1914
5.371.2.12 seekdir	1914
5.371.2.13 traits_type	1914
5.371.3 Member Enumeration Documentation	1915
5.371.3.1 event	1915
5.371.4 Constructor & Destructor Documentation	1915
5.371.4.1 basic_ostream	1915
5.371.4.2 ~basic_ostream	1915
5.371.5 Member Function Documentation	1915
5.371.5.1 _M_getloc	1915
5.371.5.2 _M_write	1916
5.371.5.3 bad	1916
5.371.5.4 clear	1917
5.371.5.5 copyfmt	1917
5.371.5.6 eof	1917
5.371.5.7 exceptions	1918
5.371.5.8 exceptions	1918
5.371.5.9 fail	1919
5.371.5.10 fill	1919

5.371.5.1lfill	1920
5.371.5.12flags	1920
5.371.5.13flags	1920
5.371.5.14flush	1921
5.371.5.15getloc	1921
5.371.5.16good	1922
5.371.5.17mbue	1922
5.371.5.18nit	1922
5.371.5.19word	1923
5.371.5.20narrow	1923
5.371.5.21operator void *	1924
5.371.5.22operator!	1924
5.371.5.23operator<<	1924
5.371.5.24operator<<	1924
5.371.5.25operator<<	1925
5.371.5.26operator<<	1925
5.371.5.27operator<<	1926
5.371.5.28operator<<	1926
5.371.5.29operator<<	1926
5.371.5.30operator<<	1927
5.371.5.31operator<<	1927
5.371.5.32operator<<	1928
5.371.5.33operator<<	1928
5.371.5.34operator<<	1928
5.371.5.35operator<<	1929
5.371.5.36operator<<	1929
5.371.5.37operator<<	1929
5.371.5.38operator<<	1930
5.371.5.39operator<<	1930
5.371.5.40precision	1931
5.371.5.41precision	1931

5.371.5.42put	1931
5.371.5.43pword	1932
5.371.5.44rdbuf	1932
5.371.5.45rdbuf	1933
5.371.5.46rdstate	1933
5.371.5.47register_callback	1934
5.371.5.48seekp	1934
5.371.5.49seekp	1935
5.371.5.50setf	1935
5.371.5.51setf	1935
5.371.5.52setstate	1936
5.371.5.53sync_with_stdio	1936
5.371.5.54tellp	1937
5.371.5.55tie	1937
5.371.5.56tie	1938
5.371.5.57unsetf	1938
5.371.5.58widen	1938
5.371.5.59width	1939
5.371.5.60width	1939
5.371.5.61write	1939
5.371.5.62xalloc	1940
5.371.6 Member Data Documentation	1940
5.371.6.1 adjustfield	1940
5.371.6.2 app	1941
5.371.6.3 ate	1941
5.371.6.4 badbit	1941
5.371.6.5 basefield	1941
5.371.6.6 beg	1941
5.371.6.7 binary	1942
5.371.6.8 boolalpha	1942
5.371.6.9 cur	1942

5.371.6.10dec	1942
5.371.6.11lend	1942
5.371.6.12eofbit	1943
5.371.6.13failbit	1943
5.371.6.14fixed	1943
5.371.6.15floatfield	1943
5.371.6.16goodbit	1944
5.371.6.17hex	1944
5.371.6.18in	1944
5.371.6.19internal	1944
5.371.6.20left	1945
5.371.6.21loct	1945
5.371.6.22out	1945
5.371.6.23right	1945
5.371.6.24scientific	1945
5.371.6.25showbase	1945
5.371.6.26showpoint	1946
5.371.6.27showpos	1946
5.371.6.28skipws	1946
5.371.6.29trunc	1946
5.371.6.30unitbuf	1946
5.371.6.31uppercase	1946
5.372std::basic_ostream< _CharT, _Traits >::sentry Class Reference . . .	1947
5.372.1 Detailed Description	1947
5.372.2 Constructor & Destructor Documentation	1947
5.372.2.1 sentry	1947
5.372.2.2 ~sentry	1948
5.372.3 Member Function Documentation	1948
5.372.3.1 operator bool	1948
5.373std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference	1948

5.373.1 Detailed Description	1954
5.373.2 Member Typedef Documentation	1955
5.373.2.1 __ctype_type	1955
5.373.2.2 __num_get_type	1955
5.373.2.3 __num_put_type	1955
5.373.2.4 char_type	1955
5.373.2.5 event_callback	1956
5.373.2.6 fmtflags	1956
5.373.2.7 int_type	1957
5.373.2.8 iostate	1957
5.373.2.9 off_type	1957
5.373.2.10 openmode	1958
5.373.2.11 pos_type	1958
5.373.2.12 seekdir	1958
5.373.2.13 traits_type	1959
5.373.3 Member Enumeration Documentation	1959
5.373.3.1 event	1959
5.373.4 Constructor & Destructor Documentation	1959
5.373.4.1 basic_ostringstream	1959
5.373.4.2 basic_ostringstream	1960
5.373.4.3 ~basic_ostringstream	1960
5.373.5 Member Function Documentation	1960
5.373.5.1 _M_getloc	1960
5.373.5.2 _M_write	1961
5.373.5.3 bad	1961
5.373.5.4 clear	1961
5.373.5.5 copyfmt	1962
5.373.5.6 eof	1962
5.373.5.7 exceptions	1963
5.373.5.8 exceptions	1963
5.373.5.9 fail	1964

5.373.5.10fill	1964
5.373.5.11fill	1965
5.373.5.12flags	1965
5.373.5.13flags	1965
5.373.5.14flush	1966
5.373.5.15getloc	1966
5.373.5.16good	1966
5.373.5.17mbue	1967
5.373.5.18nit	1967
5.373.5.19word	1967
5.373.5.20narrow	1968
5.373.5.21operator void *	1968
5.373.5.22operator!	1969
5.373.5.23operator<<	1969
5.373.5.24operator<<	1969
5.373.5.25operator<<	1970
5.373.5.26operator<<	1970
5.373.5.27operator<<	1970
5.373.5.28operator<<	1971
5.373.5.29operator<<	1971
5.373.5.30operator<<	1971
5.373.5.31operator<<	1972
5.373.5.32operator<<	1972
5.373.5.33operator<<	1973
5.373.5.34operator<<	1973
5.373.5.35operator<<	1973
5.373.5.36operator<<	1974
5.373.5.37operator<<	1974
5.373.5.38operator<<	1975
5.373.5.39operator<<	1975
5.373.5.40precision	1975

5.373.5.41precision	1976
5.373.5.42put	1976
5.373.5.43pword	1977
5.373.5.44rdbuf	1977
5.373.5.45rdbuf	1978
5.373.5.46rdstate	1978
5.373.5.47register_callback	1978
5.373.5.48seekp	1979
5.373.5.49seekp	1979
5.373.5.50setf	1980
5.373.5.51setf	1980
5.373.5.52setstate	1980
5.373.5.53str	1981
5.373.5.54str	1981
5.373.5.55sync_with_stdio	1982
5.373.5.56tellp	1982
5.373.5.57tie	1982
5.373.5.58tie	1983
5.373.5.59unsetf	1983
5.373.5.60widen	1983
5.373.5.61width	1984
5.373.5.62width	1984
5.373.5.63write	1985
5.373.5.64xalloc	1985
5.373.6 Member Data Documentation	1986
5.373.6.1 adjustfield	1986
5.373.6.2 app	1986
5.373.6.3 ate	1986
5.373.6.4 badbit	1986
5.373.6.5 basefield	1987
5.373.6.6 beg	1987

5.373.6.7 binary	1987
5.373.6.8 boolalpha	1987
5.373.6.9 cur	1987
5.373.6.10dec	1987
5.373.6.11end	1988
5.373.6.12eofbit	1988
5.373.6.13failbit	1988
5.373.6.14fixed	1988
5.373.6.15floatfield	1989
5.373.6.16goodbit	1989
5.373.6.17hex	1989
5.373.6.18n	1989
5.373.6.19internal	1990
5.373.6.20left	1990
5.373.6.21loct	1990
5.373.6.22out	1990
5.373.6.23right	1990
5.373.6.24scientific	1990
5.373.6.25showbase	1991
5.373.6.26showpoint	1991
5.373.6.27showpos	1991
5.373.6.28skipws	1991
5.373.6.29trunc	1991
5.373.6.30unitbuf	1991
5.373.6.31uppercase	1991
5.374std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	1992
5.374.1 Detailed Description	1994
5.374.2 Constructor & Destructor Documentation	1994
5.374.2.1 basic_regex	1994
5.374.2.2 basic_regex	1994
5.374.2.3 basic_regex	1995

5.374.2.4 basic_regex	1995
5.374.2.5 basic_regex	1995
5.374.2.6 basic_regex	1996
5.374.2.7 basic_regex	1996
5.374.2.8 basic_regex	1997
5.374.2.9 ~basic_regex	1997
5.374.3 Member Function Documentation	1997
5.374.3.1 assign	1997
5.374.3.2 assign	1998
5.374.3.3 assign	1998
5.374.3.4 assign	1999
5.374.3.5 assign	1999
5.374.3.6 assign	2000
5.374.3.7 assign	2000
5.374.3.8 flags	2000
5.374.3.9 getloc	2001
5.374.3.10 imbue	2001
5.374.3.11 mark_count	2001
5.374.3.12 operator=	2001
5.374.3.13 operator=	2001
5.374.3.14 operator=	2002
5.374.3.15 operator=	2002
5.374.3.16 swap	2002
5.375 std::basic_streambuf<_CharT, _Traits> Class Template Reference	2003
5.375.1 Detailed Description	2006
5.375.2 Member Typedef Documentation	2007
5.375.2.1 __streambuf_type	2007
5.375.2.2 char_type	2007
5.375.2.3 int_type	2008
5.375.2.4 off_type	2008
5.375.2.5 pos_type	2008

5.375.2.6 traits_type	2009
5.375.3 Constructor & Destructor Documentation	2009
5.375.3.1 ~basic_streambuf	2009
5.375.3.2 basic_streambuf	2009
5.375.4 Member Function Documentation	2010
5.375.4.1 eback	2010
5.375.4.2 egptr	2010
5.375.4.3 eptr	2010
5.375.4.4 gbump	2011
5.375.4.5 getloc	2011
5.375.4.6 gptr	2011
5.375.4.7 imbue	2012
5.375.4.8 in_avail	2012
5.375.4.9 overflow	2013
5.375.4.10 pbackfail	2013
5.375.4.11 pbump	2014
5.375.4.12 pptr	2015
5.375.4.13 pubimbue	2015
5.375.4.14 pubseekoff	2015
5.375.4.15 pubseekpos	2016
5.375.4.16 pubsetbuf	2016
5.375.4.17 pubsync	2016
5.375.4.18 pbumpc	2017
5.375.4.19 seekoff	2017
5.375.4.20 lseekpos	2018
5.375.4.21 setbuf	2018
5.375.4.22 setg	2018
5.375.4.23 setp	2019
5.375.4.24 sgetc	2019
5.375.4.25 getn	2019

5.375.4.27	showmanyc	2020
5.375.4.28	nextc	2020
5.375.4.29	sputbackc	2021
5.375.4.30	sputc	2021
5.375.4.31	sputn	2022
5.375.4.32	stoss	2022
5.375.4.33	sungetc	2022
5.375.4.34	sync	2023
5.375.4.35	uflow	2023
5.375.4.36	underflow	2024
5.375.4.37	xsgetn	2024
5.375.4.38	xspn	2025
5.375.5	Member Data Documentation	2025
5.375.5.1	_M_buf_locale	2025
5.375.5.2	_M_in_beg	2026
5.375.5.3	_M_in_cur	2026
5.375.5.4	_M_in_end	2026
5.375.5.5	_M_out_beg	2027
5.375.5.6	_M_out_cur	2027
5.375.5.7	_M_out_end	2027
5.376	std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference	2028
5.376.1	Detailed Description	2033
5.376.2	Constructor & Destructor Documentation	2034
5.376.2.1	basic_string	2034
5.376.2.2	basic_string	2034
5.376.2.3	basic_string	2034
5.376.2.4	basic_string	2034
5.376.2.5	basic_string	2035
5.376.2.6	basic_string	2035
5.376.2.7	basic_string	2035
5.376.2.8	basic_string	2036

5.376.2.9 basic_string	2036
5.376.2.10basic_string	2036
5.376.2.11basic_string	2037
5.376.2.12~basic_string	2037
5.376.3 Member Function Documentation	2037
5.376.3.1 append	2037
5.376.3.2 append	2038
5.376.3.3 append	2038
5.376.3.4 append	2039
5.376.3.5 append	2039
5.376.3.6 append	2039
5.376.3.7 append	2040
5.376.3.8 assign	2040
5.376.3.9 assign	2041
5.376.3.10assign	2041
5.376.3.11assign	2042
5.376.3.12assign	2042
5.376.3.13assign	2043
5.376.3.14assign	2043
5.376.3.15assign	2043
5.376.3.16at	2044
5.376.3.17at	2044
5.376.3.18back	2045
5.376.3.19back	2045
5.376.3.20begin	2045
5.376.3.21begin	2045
5.376.3.22c_str	2046
5.376.3.23capacity	2046
5.376.3.24cbegin	2046
5.376.3.25end	2046
5.376.3.26clear	2047

5.376.3.27compare	2047
5.376.3.28compare	2047
5.376.3.29compare	2048
5.376.3.30compare	2049
5.376.3.31compare	2049
5.376.3.32compare	2050
5.376.3.33copy	2051
5.376.3.34crbegin	2051
5.376.3.35crend	2051
5.376.3.36data	2052
5.376.3.37empty	2052
5.376.3.38end	2052
5.376.3.39end	2052
5.376.3.40erase	2053
5.376.3.41erase	2053
5.376.3.42erase	2054
5.376.3.43find	2054
5.376.3.44find	2055
5.376.3.45find	2055
5.376.3.46find	2055
5.376.3.47find_first_not_of	2056
5.376.3.48find_first_not_of	2056
5.376.3.49find_first_not_of	2057
5.376.3.50find_first_not_of	2057
5.376.3.51find_first_of	2058
5.376.3.52find_first_of	2058
5.376.3.53find_first_of	2059
5.376.3.54find_first_of	2059
5.376.3.55find_last_not_of	2060
5.376.3.56find_last_not_of	2060
5.376.3.57find_last_not_of	2061

5.376.3.58	find_last_not_of	2061
5.376.3.59	find_last_of	2062
5.376.3.60	find_last_of	2062
5.376.3.61	find_last_of	2063
5.376.3.62	find_last_of	2063
5.376.3.63	front	2064
5.376.3.64	front	2064
5.376.3.65	get_allocator	2064
5.376.3.66	insert	2064
5.376.3.67	insert	2065
5.376.3.68	insert	2065
5.376.3.69	insert	2066
5.376.3.70	insert	2066
5.376.3.71	insert	2067
5.376.3.72	insert	2067
5.376.3.73	insert	2068
5.376.3.74	insert	2069
5.376.3.75	length	2069
5.376.3.76	max_size	2069
5.376.3.77	operator+=	2070
5.376.3.78	operator+=	2070
5.376.3.79	operator+=	2070
5.376.3.80	operator+=	2071
5.376.3.81	operator=	2071
5.376.3.82	operator=	2071
5.376.3.83	operator=	2072
5.376.3.84	operator=	2072
5.376.3.85	operator=	2072
5.376.3.86	operator[]	2072
5.376.3.87	operator[]	2073
5.376.3.88	push_back	2073

5.376.3.89	begin	2074
5.376.3.90	begin	2074
5.376.3.91	rend	2074
5.376.3.92	rend	2074
5.376.3.93	replace	2074
5.376.3.94	replace	2075
5.376.3.95	replace	2076
5.376.3.96	replace	2076
5.376.3.97	replace	2077
5.376.3.98	replace	2077
5.376.3.99	replace	2078
5.376.3.100	place	2079
5.376.3.101	place	2079
5.376.3.102	place	2080
5.376.3.103	place	2080
5.376.3.104	reserve	2081
5.376.3.105	size	2082
5.376.3.106	size	2082
5.376.3.107	find	2083
5.376.3.108	find	2083
5.376.3.109	find	2084
5.376.3.110	find	2084
5.376.3.111	shrink_to_fit	2085
5.376.3.112	size	2085
5.376.3.113	substr	2085
5.376.3.114	swap	2086
5.376.4	Member Data Documentation	2086
5.376.4.1	npos	2086
5.377	std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference	2086
5.377.1	Detailed Description	2090

5.377.2 Member Typedef Documentation	2090
5.377.2.1 __streambuf_type	2090
5.377.2.2 char_type	2090
5.377.2.3 int_type	2091
5.377.2.4 off_type	2091
5.377.2.5 pos_type	2091
5.377.2.6 traits_type	2091
5.377.3 Constructor & Destructor Documentation	2092
5.377.3.1 basic_stringbuf	2092
5.377.3.2 basic_stringbuf	2092
5.377.4 Member Function Documentation	2092
5.377.4.1 eback	2092
5.377.4.2 egptr	2093
5.377.4.3 eptr	2093
5.377.4.4 gbump	2094
5.377.4.5 getloc	2094
5.377.4.6 gptr	2094
5.377.4.7 imbue	2095
5.377.4.8 in_avail	2095
5.377.4.9 overflow	2095
5.377.4.10 pbackfail	2096
5.377.4.11 pbase	2097
5.377.4.12 pbump	2097
5.377.4.13 pptr	2097
5.377.4.14 pubimbue	2098
5.377.4.15 pubseekoff	2098
5.377.4.16 pubseekpos	2099
5.377.4.17 pubsetbuf	2099
5.377.4.18 pubsync	2099
5.377.4.19 sbumpc	2100
5.377.4.20 seekoff	2100

5.377.4.21seekpos	2100
5.377.4.22setbuf	2101
5.377.4.23setbuf	2101
5.377.4.24setg	2102
5.377.4.25setp	2102
5.377.4.26sgetc	2102
5.377.4.27sgetn	2103
5.377.4.28showmanyc	2103
5.377.4.29nextc	2104
5.377.4.30sputbackc	2104
5.377.4.31sputc	2104
5.377.4.32sputn	2105
5.377.4.33stossc	2105
5.377.4.34str	2106
5.377.4.35str	2106
5.377.4.36sungetc	2106
5.377.4.37sync	2107
5.377.4.38uflow	2107
5.377.4.39underflow	2108
5.377.4.40xsgetn	2108
5.377.4.41xsputn	2109
5.377.5 Member Data Documentation	2110
5.377.5.1 _M_buf_locale	2110
5.377.5.2 _M_in_beg	2110
5.377.5.3 _M_in_cur	2110
5.377.5.4 _M_in_end	2110
5.377.5.5 _M_mode	2111
5.377.5.6 _M_out_beg	2111
5.377.5.7 _M_out_cur	2111
5.377.5.8 _M_out_end	2112

5.378std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template	
Reference	2112
5.378.1 Detailed Description	2120
5.378.2 Member Typedef Documentation	2121
5.378.2.1 __ctype_type	2121
5.378.2.2 __ctype_type	2121
5.378.2.3 __num_get_type	2121
5.378.2.4 __num_put_type	2121
5.378.2.5 __num_put_type	2122
5.378.2.6 char_type	2122
5.378.2.7 char_type	2122
5.378.2.8 event_callback	2122
5.378.2.9 fmtflags	2123
5.378.2.10int_type	2124
5.378.2.11int_type	2124
5.378.2.12iostate	2124
5.378.2.13off_type	2125
5.378.2.14off_type	2125
5.378.2.15openmode	2125
5.378.2.16pos_type	2126
5.378.2.17pos_type	2126
5.378.2.18seekdir	2126
5.378.2.19traits_type	2126
5.378.2.20traits_type	2127
5.378.3 Member Enumeration Documentation	2127
5.378.3.1 event	2127
5.378.4 Constructor & Destructor Documentation	2127
5.378.4.1 basic_stringstream	2127
5.378.4.2 basic_stringstream	2128
5.378.4.3 ~basic_stringstream	2128
5.378.5 Member Function Documentation	2128

5.378.5.1 _M_getloc	2128
5.378.5.2 _M_write	2129
5.378.5.3 bad	2129
5.378.5.4 clear	2130
5.378.5.5 copyfmt	2130
5.378.5.6 eof	2130
5.378.5.7 exceptions	2131
5.378.5.8 exceptions	2131
5.378.5.9 fail	2132
5.378.5.10fill	2132
5.378.5.11fill	2133
5.378.5.12flags	2133
5.378.5.13flags	2133
5.378.5.14flush	2134
5.378.5.15gcount	2134
5.378.5.16get	2134
5.378.5.17get	2135
5.378.5.18get	2135
5.378.5.19get	2136
5.378.5.20get	2137
5.378.5.21get	2137
5.378.5.22getline	2138
5.378.5.23getline	2139
5.378.5.24getloc	2139
5.378.5.25good	2140
5.378.5.26ignore	2140
5.378.5.27ignore	2140
5.378.5.28ignore	2141
5.378.5.29mbue	2141
5.378.5.30nit	2142
5.378.5.31word	2142

5.378.5.32	narrow	2143
5.378.5.33	operator void *	2143
5.378.5.34	operator!	2143
5.378.5.35	operator<<	2144
5.378.5.36	operator<<	2144
5.378.5.37	operator<<	2144
5.378.5.38	operator<<	2145
5.378.5.39	operator<<	2145
5.378.5.40	operator<<	2145
5.378.5.41	operator<<	2146
5.378.5.42	operator<<	2146
5.378.5.43	operator<<	2146
5.378.5.44	operator<<	2147
5.378.5.45	operator<<	2147
5.378.5.46	operator<<	2148
5.378.5.47	operator<<	2148
5.378.5.48	operator<<	2148
5.378.5.49	operator<<	2149
5.378.5.50	operator<<	2150
5.378.5.51	operator<<	2150
5.378.5.52	operator>>	2150
5.378.5.53	operator>>	2151
5.378.5.54	operator>>	2151
5.378.5.55	operator>>	2152
5.378.5.56	operator>>	2152
5.378.5.57	operator>>	2152
5.378.5.58	operator>>	2153
5.378.5.59	operator>>	2153
5.378.5.60	operator>>	2153
5.378.5.61	operator>>	2154
5.378.5.62	operator>>	2154

5.378.5.63operator>>	2155
5.378.5.64operator>>	2155
5.378.5.65operator>>	2156
5.378.5.66operator>>	2156
5.378.5.67operator>>	2156
5.378.5.68operator>>	2156
5.378.5.69peek	2157
5.378.5.70precision	2157
5.378.5.71precision	2158
5.378.5.72put	2158
5.378.5.73putback	2158
5.378.5.74pword	2159
5.378.5.75rdbuf	2160
5.378.5.76rdbuf	2160
5.378.5.77rdstate	2161
5.378.5.78read	2161
5.378.5.79readsome	2162
5.378.5.80register_callback	2162
5.378.5.81seekg	2163
5.378.5.82seekg	2163
5.378.5.83seekp	2164
5.378.5.84seekp	2164
5.378.5.85setf	2165
5.378.5.86setf	2165
5.378.5.87setstate	2166
5.378.5.88str	2166
5.378.5.89str	2166
5.378.5.90sync	2167
5.378.5.91sync_with_stdio	2167
5.378.5.92tellg	2168
5.378.5.93tellp	2168

5.378.5.94	tie	2168
5.378.5.95	tie	2169
5.378.5.96	unget	2169
5.378.5.97	unsetf	2170
5.378.5.98	widen	2170
5.378.5.99	width	2171
5.378.5.100	width	2171
5.378.5.101	write	2171
5.378.5.102	zalloc	2172
5.378.6	Member Data Documentation	2172
5.378.6.1	_M_gcount	2172
5.378.6.2	adjustfield	2172
5.378.6.3	app	2173
5.378.6.4	ate	2173
5.378.6.5	badbit	2173
5.378.6.6	basefield	2173
5.378.6.7	beg	2174
5.378.6.8	binary	2174
5.378.6.9	boolalpha	2174
5.378.6.10	cur	2174
5.378.6.11	dec	2174
5.378.6.12	end	2174
5.378.6.13	eofbit	2175
5.378.6.14	failbit	2175
5.378.6.15	fixed	2175
5.378.6.16	floatfield	2175
5.378.6.17	goodbit	2176
5.378.6.18	hex	2176
5.378.6.19	in	2176
5.378.6.20	internal	2176
5.378.6.21	left	2177

5.378.6.22oct	2177
5.378.6.23out	2177
5.378.6.24right	2177
5.378.6.25scientific	2177
5.378.6.26showbase	2177
5.378.6.27showpoint	2178
5.378.6.28showpos	2178
5.378.6.29skipws	2178
5.378.6.30trunc	2178
5.378.6.31unitbuf	2178
5.378.6.32uppercase	2178
5.379std::bernoulli_distribution Class Reference	2179
5.379.1 Detailed Description	2179
5.379.2 Member Typedef Documentation	2179
5.379.2.1 result_type	2179
5.379.3 Constructor & Destructor Documentation	2180
5.379.3.1 bernoulli_distribution	2180
5.379.4 Member Function Documentation	2180
5.379.4.1 max	2180
5.379.4.2 min	2180
5.379.4.3 operator()	2180
5.379.4.4 p	2180
5.379.4.5 param	2181
5.379.4.6 param	2181
5.379.4.7 reset	2181
5.380std::bernoulli_distribution::param_type Struct Reference	2181
5.380.1 Detailed Description	2182
5.381std::bidirectional_iterator_tag Struct Reference	2182
5.381.1 Detailed Description	2183
5.382std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference	2183

5.382.1 Detailed Description	2184
5.382.2 Member Typedef Documentation	2184
5.382.2.1 first_argument_type	2184
5.382.2.2 result_type	2184
5.382.2.3 second_argument_type	2184
5.383std::binary_negate< _Predicate > Class Template Reference	2185
5.383.1 Detailed Description	2185
5.383.2 Member Typedef Documentation	2185
5.383.2.1 first_argument_type	2185
5.383.2.2 result_type	2186
5.383.2.3 second_argument_type	2186
5.384std::binder1st< _Operation > Class Template Reference	2186
5.384.1 Detailed Description	2187
5.384.2 Member Typedef Documentation	2187
5.384.2.1 argument_type	2187
5.384.2.2 result_type	2187
5.385std::binder2nd< _Operation > Class Template Reference	2188
5.385.1 Detailed Description	2189
5.385.2 Member Typedef Documentation	2189
5.385.2.1 argument_type	2189
5.385.2.2 result_type	2189
5.386std::binomial_distribution< _IntType > Class Template Reference	2189
5.386.1 Detailed Description	2190
5.386.2 Member Typedef Documentation	2191
5.386.2.1 result_type	2191
5.386.3 Member Function Documentation	2191
5.386.3.1 max	2191
5.386.3.2 min	2191
5.386.3.3 operator()	2191
5.386.3.4 operator()	2192
5.386.3.5 p	2192

5.386.3.6 param	2192
5.386.3.7 param	2192
5.386.3.8 reset	2193
5.386.3.9 t	2193
5.386.4 Friends And Related Function Documentation	2193
5.386.4.1 operator<<	2193
5.386.4.2 operator==	2193
5.386.4.3 operator>>	2194
5.387std::binomial_distribution< _IntType >::param_type Struct Reference	2194
5.387.1 Detailed Description	2195
5.388std::bitset< _Nb > Class Template Reference	2195
5.388.1 Detailed Description	2198
5.388.2 Constructor & Destructor Documentation	2200
5.388.2.1 bitset	2200
5.388.2.2 bitset	2200
5.388.2.3 bitset	2200
5.388.2.4 bitset	2200
5.388.2.5 bitset	2201
5.388.3 Member Function Documentation	2201
5.388.3.1 all	2201
5.388.3.2 any	2201
5.388.3.3 count	2202
5.388.3.4 flip	2202
5.388.3.5 flip	2202
5.388.3.6 none	2202
5.388.3.7 operator!=	2203
5.388.3.8 operator&=	2203
5.388.3.9 operator<<	2203
5.388.3.10operator<<=	2203
5.388.3.11operator==	2203
5.388.3.12operator>>	2204

5.388.3.13	<code>operator>>=</code>	2204
5.388.3.14	<code>operator[]</code>	2204
5.388.3.15	<code>operator[]</code>	2205
5.388.3.16	<code>operator^=</code>	2205
5.388.3.17	<code>operator =</code>	2205
5.388.3.18	<code>operator~</code>	2206
5.388.3.19	<code>reset</code>	2206
5.388.3.20	<code>reset</code>	2206
5.388.3.21	<code>set</code>	2206
5.388.3.22	<code>set</code>	2206
5.388.3.23	<code>size</code>	2207
5.388.3.24	<code>test</code>	2207
5.388.3.25	<code>to_string</code>	2207
5.388.3.26	<code>to_ulong</code>	2208
5.389	<code>std::bitset< _Nb >::reference</code> Class Reference	2208
5.389.1	Detailed Description	2209
5.390	<code>std::cauchy_distribution< _RealType ></code> Class Template Reference	2209
5.390.1	Detailed Description	2210
5.390.2	Member Typedef Documentation	2210
5.390.2.1	<code>result_type</code>	2210
5.390.3	Member Function Documentation	2210
5.390.3.1	<code>max</code>	2210
5.390.3.2	<code>min</code>	2210
5.390.3.3	<code>operator()</code>	2211
5.390.3.4	<code>param</code>	2211
5.390.3.5	<code>param</code>	2211
5.390.3.6	<code>reset</code>	2211
5.391	<code>std::cauchy_distribution< _RealType >::param_type</code> Struct Reference	2212
5.391.1	Detailed Description	2212
5.392	<code>std::char_traits< _CharT ></code> Struct Template Reference	2212
5.392.1	Detailed Description	2214

5.393std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference	2214
5.393.1 Detailed Description	2215
5.394std::char_traits< char > Struct Template Reference	2215
5.394.1 Detailed Description	2216
5.395std::char_traits< wchar_t > Struct Template Reference	2216
5.395.1 Detailed Description	2217
5.396std::chi_squared_distribution< _RealType > Class Template Reference	2217
5.396.1 Detailed Description	2218
5.396.2 Member Typedef Documentation	2218
5.396.2.1 result_type	2218
5.396.3 Member Function Documentation	2219
5.396.3.1 max	2219
5.396.3.2 min	2219
5.396.3.3 operator()	2219
5.396.3.4 param	2219
5.396.3.5 param	2219
5.396.3.6 reset	2220
5.396.4 Friends And Related Function Documentation	2220
5.396.4.1 operator<<	2220
5.396.4.2 operator==	2220
5.396.4.3 operator>>	2221
5.397std::chi_squared_distribution< _RealType >::param_type Struct Reference	2221
5.397.1 Detailed Description	2222
5.398std::chrono::duration< _Rep, _Period > Struct Template Reference	2222
5.398.1 Detailed Description	2223
5.399std::chrono::duration_values< _Rep > Struct Template Reference	2223
5.399.1 Detailed Description	2224
5.400std::chrono::system_clock Struct Reference	2224
5.400.1 Detailed Description	2225

5.401 <code>std::chrono::time_point< _Clock, _Duration ></code> Struct Template Reference	2225
5.401.1 Detailed Description	2225
5.402 <code>std::chrono::treat_as_floating_point< _Rep ></code> Struct Template Reference	2226
5.402.1 Detailed Description	2226
5.403 <code>std::codecvt< _InternT, _ExternT, _StateT ></code> Class Template Reference	2226
5.403.1 Detailed Description	2229
5.403.2 Member Function Documentation	2229
5.403.2.1 <code>do_out</code>	2229
5.403.2.2 <code>in</code>	2229
5.403.2.3 <code>out</code>	2230
5.403.2.4 <code>unshift</code>	2231
5.404 <code>std::codecvt< _InternT, _ExternT, encoding_state ></code> Class Template Reference	2232
5.404.1 Detailed Description	2234
5.404.2 Member Function Documentation	2235
5.404.2.1 <code>do_out</code>	2235
5.404.2.2 <code>in</code>	2235
5.404.2.3 <code>out</code>	2236
5.404.2.4 <code>unshift</code>	2237
5.405 <code>std::codecvt< char, char, mbstate_t ></code> Class Template Reference	2238
5.405.1 Detailed Description	2240
5.405.2 Member Function Documentation	2240
5.405.2.1 <code>do_out</code>	2240
5.405.2.2 <code>in</code>	2240
5.405.2.3 <code>out</code>	2241
5.405.2.4 <code>unshift</code>	2242
5.406 <code>std::codecvt< wchar_t, char, mbstate_t ></code> Class Template Reference	2243
5.406.1 Detailed Description	2245
5.406.2 Member Function Documentation	2245
5.406.2.1 <code>do_out</code>	2245
5.406.2.2 <code>in</code>	2246

5.406.2.3 out	2247
5.406.2.4 unshift	2248
5.407std::codecvt_base Class Reference	2248
5.407.1 Detailed Description	2249
5.408std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	2249
5.408.1 Detailed Description	2252
5.408.2 Member Function Documentation	2252
5.408.2.1 do_out	2252
5.408.2.2 in	2252
5.408.2.3 out	2253
5.408.2.4 unshift	2254
5.409std::collate< _CharT > Class Template Reference	2255
5.409.1 Detailed Description	2257
5.409.2 Member Typedef Documentation	2258
5.409.2.1 char_type	2258
5.409.2.2 string_type	2258
5.409.3 Constructor & Destructor Documentation	2258
5.409.3.1 collate	2258
5.409.3.2 collate	2258
5.409.3.3 ~collate	2259
5.409.4 Member Function Documentation	2259
5.409.4.1 compare	2259
5.409.4.2 do_compare	2259
5.409.4.3 do_hash	2260
5.409.4.4 do_transform	2260
5.409.4.5 hash	2261
5.409.4.6 transform	2261
5.409.5 Member Data Documentation	2262
5.409.5.1 id	2262
5.410std::collate_byname< _CharT > Class Template Reference	2262

5.410.1 Detailed Description	2264
5.410.2 Member Typedef Documentation	2265
5.410.2.1 char_type	2265
5.410.2.2 string_type	2265
5.410.3 Member Function Documentation	2265
5.410.3.1 compare	2265
5.410.3.2 do_compare	2266
5.410.3.3 do_hash	2266
5.410.3.4 do_transform	2267
5.410.3.5 hash	2267
5.410.3.6 transform	2268
5.410.4 Member Data Documentation	2268
5.410.4.1 id	2268
5.411 std::complex< _Tp > Struct Template Reference	2268
5.411.1 Detailed Description	2269
5.411.2 Member Typedef Documentation	2270
5.411.2.1 value_type	2270
5.411.3 Constructor & Destructor Documentation	2270
5.411.3.1 complex	2270
5.411.3.2 complex	2270
5.411.4 Member Function Documentation	2270
5.411.4.1 operator+=	2270
5.411.4.2 operator-=	2270
5.412 std::condition_variable Class Reference	2271
5.412.1 Detailed Description	2271
5.413 std::condition_variable_any Class Reference	2272
5.413.1 Detailed Description	2272
5.414 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference	2273
5.414.1 Detailed Description	2273
5.415 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2273

5.415.1 Detailed Description	2274
5.415.2 Member Typedef Documentation	2275
5.415.2.1 first_argument_type	2275
5.415.2.2 result_type	2275
5.415.2.3 second_argument_type	2275
5.416std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2275
5.416.1 Detailed Description	2276
5.416.2 Member Typedef Documentation	2277
5.416.2.1 first_argument_type	2277
5.416.2.2 result_type	2277
5.416.2.3 second_argument_type	2277
5.417std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference .	2277
5.417.1 Detailed Description	2278
5.417.2 Member Typedef Documentation	2279
5.417.2.1 argument_type	2279
5.417.2.2 result_type	2279
5.418std::const_mem_fun_t< _Ret, _Tp > Class Template Reference . . .	2279
5.418.1 Detailed Description	2280
5.418.2 Member Typedef Documentation	2281
5.418.2.1 argument_type	2281
5.418.2.2 result_type	2281
5.419std::ctype< _CharT > Class Template Reference	2281
5.419.1 Detailed Description	2284
5.419.2 Member Typedef Documentation	2284
5.419.2.1 char_type	2284
5.419.3 Member Function Documentation	2285
5.419.3.1 do_is	2285
5.419.3.2 do_is	2285
5.419.3.3 do_narrow	2286
5.419.3.4 do_narrow	2286
5.419.3.5 do_scan_is	2287

5.419.3.6 do_scan_not	2287
5.419.3.7 do_tolower	2288
5.419.3.8 do_tolower	2288
5.419.3.9 do_toupper	2289
5.419.3.10 do_toupper	2289
5.419.3.11 do_widen	2290
5.419.3.12 do_widen	2290
5.419.3.13 is	2291
5.419.3.14 is	2291
5.419.3.15 narrow	2292
5.419.3.16 narrow	2292
5.419.3.17 scan_is	2293
5.419.3.18 scan_not	2293
5.419.3.19 tolower	2294
5.419.3.20 tolower	2294
5.419.3.21 toupper	2294
5.419.3.22 toupper	2295
5.419.3.23 widen	2295
5.419.3.24 widen	2296
5.419.4 Member Data Documentation	2296
5.419.4.1 id	2296
5.420 std::ctype< char > Class Template Reference	2297
5.420.1 Detailed Description	2299
5.420.2 Member Typedef Documentation	2300
5.420.2.1 char_type	2300
5.420.3 Constructor & Destructor Documentation	2300
5.420.3.1 ctype	2300
5.420.3.2 ctype	2300
5.420.3.3 ~ctype	2300
5.420.4 Member Function Documentation	2301
5.420.4.1 classic_table	2301

5.420.4.2 do_narrow	2301
5.420.4.3 do_narrow	2301
5.420.4.4 do_tolower	2302
5.420.4.5 do_tolower	2302
5.420.4.6 do_toupper	2303
5.420.4.7 do_toupper	2303
5.420.4.8 do_widen	2304
5.420.4.9 do_widen	2304
5.420.4.10 is	2305
5.420.4.11 is	2305
5.420.4.12 narrow	2305
5.420.4.13 narrow	2306
5.420.4.14 scan_is	2307
5.420.4.15 scan_not	2307
5.420.4.16 table	2307
5.420.4.17 tolower	2308
5.420.4.18 tolower	2308
5.420.4.19 toupper	2308
5.420.4.20 toupper	2309
5.420.4.21 widen	2309
5.420.4.22 widen	2310
5.420.5 Member Data Documentation	2310
5.420.5.1 id	2310
5.420.5.2 table_size	2310
5.421 std::ctype< wchar_t > Class Template Reference	2311
5.421.1 Detailed Description	2314
5.421.2 Member Typedef Documentation	2314
5.421.2.1 char_type	2314
5.421.3 Constructor & Destructor Documentation	2315
5.421.3.1 ctype	2315
5.421.3.2 ctype	2315

5.421.3.3 ~ctype	2315
5.421.4 Member Function Documentation	2315
5.421.4.1 do_is	2315
5.421.4.2 do_is	2316
5.421.4.3 do_is	2316
5.421.4.4 do_is	2317
5.421.4.5 do_narrow	2317
5.421.4.6 do_narrow	2318
5.421.4.7 do_narrow	2318
5.421.4.8 do_narrow	2319
5.421.4.9 do_scan_is	2319
5.421.4.10do_scan_is	2320
5.421.4.11do_scan_not	2320
5.421.4.12do_scan_not	2321
5.421.4.13do_tolower	2321
5.421.4.14do_tolower	2321
5.421.4.15do_tolower	2322
5.421.4.16do_tolower	2322
5.421.4.17do_toupper	2323
5.421.4.18do_toupper	2323
5.421.4.19do_toupper	2324
5.421.4.20do_toupper	2324
5.421.4.21do_widen	2324
5.421.4.22do_widen	2325
5.421.4.23do_widen	2325
5.421.4.24is	2326
5.421.4.25is	2326
5.421.4.26narrow	2327
5.421.4.27narrow	2327
5.421.4.28scan_is	2328
5.421.4.29scan_not	2328

5.421.4.30	tolower	2329
5.421.4.31	tolower	2329
5.421.4.32	toupper	2330
5.421.4.33	toupper	2330
5.421.4.34	widen	2330
5.421.4.35	widen	2331
5.421.5	Member Data Documentation	2331
5.421.5.1	id	2331
5.422	std::ctype_base Struct Reference	2332
5.422.1	Detailed Description	2332
5.423	std::ctype_byname<_CharT> Class Template Reference	2333
5.423.1	Detailed Description	2335
5.423.2	Member Typedef Documentation	2335
5.423.2.1	char_type	2335
5.423.3	Member Function Documentation	2336
5.423.3.1	do_is	2336
5.423.3.2	do_is	2336
5.423.3.3	do_narrow	2337
5.423.3.4	do_narrow	2337
5.423.3.5	do_scan_is	2338
5.423.3.6	do_scan_not	2338
5.423.3.7	do_tolower	2339
5.423.3.8	do_tolower	2339
5.423.3.9	do_toupper	2340
5.423.3.10	do_toupper	2340
5.423.3.11	do_widen	2341
5.423.3.12	do_widen	2341
5.423.3.13	is	2342
5.423.3.14	is	2342
5.423.3.15	narrow	2343
5.423.3.16	narrow	2343

5.423.3.17	scan_is	2344
5.423.3.18	scan_not	2344
5.423.3.19	tolower	2345
5.423.3.20	tolower	2345
5.423.3.21	toupper	2345
5.423.3.22	toupper	2346
5.423.3.23	widen	2346
5.423.3.24	widen	2347
5.423.4	Member Data Documentation	2347
5.423.4.1	id	2347
5.424	std::ctype_byname< char > Class Template Reference	2348
5.424.1	Detailed Description	2350
5.424.2	Member Typedef Documentation	2350
5.424.2.1	char_type	2350
5.424.3	Member Function Documentation	2351
5.424.3.1	classic_table	2351
5.424.3.2	do_narrow	2351
5.424.3.3	do_narrow	2351
5.424.3.4	do_tolower	2352
5.424.3.5	do_tolower	2352
5.424.3.6	do_toupper	2353
5.424.3.7	do_toupper	2353
5.424.3.8	do_widen	2354
5.424.3.9	do_widen	2354
5.424.3.10	is	2355
5.424.3.11	is	2355
5.424.3.12	narrow	2355
5.424.3.13	narrow	2356
5.424.3.14	scan_is	2357
5.424.3.15	scan_not	2357
5.424.3.16	table	2357

5.424.3.17 tolower	2358
5.424.3.18 tolower	2358
5.424.3.19 toupper	2358
5.424.3.20 toupper	2359
5.424.3.21 widen	2359
5.424.3.22 widen	2360
5.424.4 Member Data Documentation	2360
5.424.4.1 id	2360
5.424.4.2 table_size	2360
5.425 std::decay< _Tp > Class Template Reference	2361
5.425.1 Detailed Description	2361
5.426 std::decimal::decimal128 Class Reference	2361
5.426.1 Detailed Description	2363
5.426.2 Constructor & Destructor Documentation	2363
5.426.2.1 decimal128	2363
5.427 std::decimal::decimal32 Class Reference	2363
5.427.1 Detailed Description	2365
5.427.2 Constructor & Destructor Documentation	2365
5.427.2.1 decimal32	2365
5.428 std::decimal::decimal64 Class Reference	2365
5.428.1 Detailed Description	2367
5.428.2 Constructor & Destructor Documentation	2367
5.428.2.1 decimal64	2367
5.429 std::default_delete< _Tp > Struct Template Reference	2367
5.429.1 Detailed Description	2368
5.430 std::default_delete< _Tp[]> Struct Template Reference	2368
5.430.1 Detailed Description	2368
5.431 std::defer_lock_t Struct Reference	2368
5.431.1 Detailed Description	2368
5.432 std::deque< _Tp, _Alloc > Class Template Reference	2369
5.432.1 Detailed Description	2374

5.432.2 Constructor & Destructor Documentation	2375
5.432.2.1 deque	2375
5.432.2.2 deque	2375
5.432.2.3 deque	2376
5.432.2.4 deque	2376
5.432.2.5 deque	2376
5.432.2.6 deque	2377
5.432.2.7 deque	2377
5.432.2.8 deque	2378
5.432.2.9 ~deque	2378
5.432.3 Member Function Documentation	2378
5.432.3.1 _M_fill_initialize	2378
5.432.3.2 _M_initialize_map	2379
5.432.3.3 _M_new_elements_at_back	2379
5.432.3.4 _M_new_elements_at_front	2380
5.432.3.5 _M_pop_back_aux	2380
5.432.3.6 _M_pop_front_aux	2380
5.432.3.7 _M_push_back_aux	2380
5.432.3.8 _M_push_front_aux	2380
5.432.3.9 _M_range_check	2381
5.432.3.10 _M_range_initialize	2381
5.432.3.11 _M_range_initialize	2381
5.432.3.12 _M_reallocate_map	2382
5.432.3.13 _M_reserve_elements_at_back	2382
5.432.3.14 _M_reserve_elements_at_front	2382
5.432.3.15 _M_reserve_map_at_back	2383
5.432.3.16 _M_reserve_map_at_front	2383
5.432.3.17 assign	2383
5.432.3.18 assign	2384
5.432.3.19 assign	2384
5.432.3.20 at	2384

5.432.3.2lat	2385
5.432.3.22back	2385
5.432.3.23back	2386
5.432.3.24begin	2386
5.432.3.25begin	2386
5.432.3.26cbegin	2386
5.432.3.27cend	2387
5.432.3.28clear	2387
5.432.3.29crbegin	2387
5.432.3.30crend	2387
5.432.3.31emplace	2387
5.432.3.32empty	2388
5.432.3.33end	2388
5.432.3.34end	2388
5.432.3.35erase	2389
5.432.3.36erase	2389
5.432.3.37front	2390
5.432.3.38front	2390
5.432.3.39get_allocator	2390
5.432.3.40insert	2390
5.432.3.41insert	2391
5.432.3.42insert	2391
5.432.3.43insert	2391
5.432.3.44insert	2392
5.432.3.45max_size	2392
5.432.3.46operator=	2392
5.432.3.47operator=	2393
5.432.3.48operator=	2393
5.432.3.49operator[]	2394
5.432.3.50operator[]	2394
5.432.3.51pop_back	2394

5.432.3.52	pop_front	2395
5.432.3.53	push_back	2395
5.432.3.54	push_front	2395
5.432.3.55	rbegin	2396
5.432.3.56	rbegin	2396
5.432.3.57	rend	2396
5.432.3.58	rend	2396
5.432.3.59	resize	2397
5.432.3.60	resize	2397
5.432.3.61	shrink_to_fit	2397
5.432.3.62	size	2398
5.432.3.63	swap	2398
5.433	std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference	2398
5.433.1	Detailed Description	2400
5.433.2	Member Typedef Documentation	2400
5.433.2.1	result_type	2400
5.433.3	Constructor & Destructor Documentation	2400
5.433.3.1	discard_block_engine	2400
5.433.3.2	discard_block_engine	2400
5.433.3.3	discard_block_engine	2401
5.433.3.4	discard_block_engine	2401
5.433.3.5	discard_block_engine	2401
5.433.4	Member Function Documentation	2402
5.433.4.1	base	2402
5.433.4.2	discard	2402
5.433.4.3	max	2402
5.433.4.4	min	2402
5.433.4.5	operator()	2403
5.433.4.6	seed	2403
5.433.4.7	seed	2403

5.433.4.8 seed	2403
5.433.5 Friends And Related Function Documentation	2404
5.433.5.1 operator<<	2404
5.433.5.2 operator==	2404
5.433.5.3 operator>>	2405
5.434std::discrete_distribution< _IntType > Class Template Reference . .	2405
5.434.1 Detailed Description	2406
5.434.2 Member Typedef Documentation	2406
5.434.2.1 result_type	2406
5.434.3 Member Function Documentation	2407
5.434.3.1 max	2407
5.434.3.2 min	2407
5.434.3.3 operator()	2407
5.434.3.4 param	2407
5.434.3.5 param	2408
5.434.3.6 probabilities	2408
5.434.3.7 reset	2408
5.434.4 Friends And Related Function Documentation	2408
5.434.4.1 operator<<	2408
5.434.4.2 operator>>	2409
5.435std::discrete_distribution< _IntType >::param_type Struct Reference	2409
5.435.1 Detailed Description	2410
5.436std::divides< _Tp > Struct Template Reference	2410
5.436.1 Detailed Description	2411
5.436.2 Member Typedef Documentation	2411
5.436.2.1 first_argument_type	2411
5.436.2.2 result_type	2411
5.436.2.3 second_argument_type	2411
5.437std::domain_error Class Reference	2412
5.437.1 Detailed Description	2412
5.437.2 Member Function Documentation	2412

5.437.2.1 what	2412
5.438std::enable_if< bool, _Tp > Struct Template Reference	2413
5.438.1 Detailed Description	2413
5.439std::enable_shared_from_this< _Tp > Class Template Reference	2413
5.439.1 Detailed Description	2414
5.440std::equal_to< _Tp > Struct Template Reference	2414
5.440.1 Detailed Description	2415
5.440.2 Member Typedef Documentation	2416
5.440.2.1 first_argument_type	2416
5.440.2.2 result_type	2416
5.440.2.3 second_argument_type	2416
5.441std::error_category Class Reference	2416
5.441.1 Detailed Description	2417
5.442std::error_code Struct Reference	2417
5.442.1 Detailed Description	2417
5.443std::error_condition Struct Reference	2418
5.443.1 Detailed Description	2418
5.444std::exception Class Reference	2418
5.444.1 Detailed Description	2420
5.444.2 Member Function Documentation	2420
5.444.2.1 what	2420
5.445std::exponential_distribution< _RealType > Class Template Reference	2420
5.445.1 Detailed Description	2421
5.445.2 Member Typedef Documentation	2421
5.445.2.1 result_type	2421
5.445.3 Constructor & Destructor Documentation	2422
5.445.3.1 exponential_distribution	2422
5.445.4 Member Function Documentation	2422
5.445.4.1 lambda	2422
5.445.4.2 max	2422
5.445.4.3 min	2422

5.445.4.4 operator()	2422
5.445.4.5 param	2423
5.445.4.6 param	2423
5.445.4.7 reset	2423
5.446std::exponential_distribution< _RealType >::param_type Struct Reference	2424
5.446.1 Detailed Description	2424
5.447std::extreme_value_distribution< _RealType > Class Template Reference	2424
5.447.1 Detailed Description	2425
5.447.2 Member Typedef Documentation	2425
5.447.2.1 result_type	2425
5.447.3 Member Function Documentation	2426
5.447.3.1 a	2426
5.447.3.2 b	2426
5.447.3.3 max	2426
5.447.3.4 min	2426
5.447.3.5 operator()	2426
5.447.3.6 param	2427
5.447.3.7 param	2427
5.447.3.8 reset	2427
5.448std::extreme_value_distribution< _RealType >::param_type Struct Reference	2427
5.448.1 Detailed Description	2428
5.449std::fisher_f_distribution< _RealType > Class Template Reference	2428
5.449.1 Detailed Description	2429
5.449.2 Member Typedef Documentation	2429
5.449.2.1 result_type	2429
5.449.3 Member Function Documentation	2430
5.449.3.1 max	2430
5.449.3.2 min	2430
5.449.3.3 operator()	2430

5.449.3.4 param	2430
5.449.3.5 param	2430
5.449.3.6 reset	2431
5.449.4 Friends And Related Function Documentation	2431
5.449.4.1 operator<<	2431
5.449.4.2 operator==	2431
5.449.4.3 operator>>	2432
5.450std::fisher_f_distribution< _RealType >::param_type Struct Reference	2432
5.450.1 Detailed Description	2433
5.451std::forward_iterator_tag Struct Reference	2433
5.451.1 Detailed Description	2434
5.452std::forward_list< _Tp, _Alloc > Class Template Reference	2434
5.452.1 Detailed Description	2437
5.452.2 Constructor & Destructor Documentation	2438
5.452.2.1 forward_list	2438
5.452.2.2 forward_list	2438
5.452.2.3 forward_list	2438
5.452.2.4 forward_list	2438
5.452.2.5 forward_list	2439
5.452.2.6 forward_list	2439
5.452.2.7 forward_list	2440
5.452.2.8 forward_list	2440
5.452.2.9 forward_list	2440
5.452.2.10~forward_list	2441
5.452.3 Member Function Documentation	2441
5.452.3.1 assign	2441
5.452.3.2 assign	2441
5.452.3.3 assign	2442
5.452.3.4 before_begin	2442
5.452.3.5 before_begin	2442
5.452.3.6 begin	2442

5.452.3.7 begin	2443
5.452.3.8 cbefore_begin	2443
5.452.3.9 cbegin	2443
5.452.3.10 cend	2443
5.452.3.11 clear	2443
5.452.3.12 emplace_after	2444
5.452.3.13 emplace_front	2444
5.452.3.14 empty	2445
5.452.3.15 end	2445
5.452.3.16 end	2445
5.452.3.17 erase_after	2445
5.452.3.18 erase_after	2446
5.452.3.19 front	2446
5.452.3.20 front	2446
5.452.3.21 get_allocator	2447
5.452.3.22 insert_after	2447
5.452.3.23 insert_after	2447
5.452.3.24 insert_after	2448
5.452.3.25 insert_after	2448
5.452.3.26 max_size	2449
5.452.3.27 merge	2449
5.452.3.28 merge	2449
5.452.3.29 operator=	2450
5.452.3.30 operator=	2450
5.452.3.31 operator=	2450
5.452.3.32 pop_front	2451
5.452.3.33 push_front	2451
5.452.3.34 remove	2451
5.452.3.35 remove_if	2452
5.452.3.36 resize	2452
5.452.3.37 resize	2452

5.452.3.38	reverse	2453
5.452.3.39	sort	2453
5.452.3.40	sort	2453
5.452.3.41	splice_after	2454
5.452.3.42	splice_after	2454
5.452.3.43	splice_after	2454
5.452.3.44	swap	2455
5.452.3.45	unique	2455
5.452.3.46	unique	2455
5.453	std::fpos< _StateT > Class Template Reference	2456
5.453.1	Detailed Description	2456
5.453.2	Constructor & Destructor Documentation	2457
5.453.2.1	fpos	2457
5.453.3	Member Function Documentation	2457
5.453.3.1	operator streamoff	2457
5.453.3.2	operator+	2457
5.453.3.3	operator+=	2457
5.453.3.4	operator-	2457
5.453.3.5	operator-	2458
5.453.3.6	operator-=	2458
5.453.3.7	state	2458
5.453.3.8	state	2458
5.454	std::front_insert_iterator< _Container > Class Template Reference	2458
5.454.1	Detailed Description	2460
5.454.2	Member Typedef Documentation	2460
5.454.2.1	container_type	2460
5.454.2.2	difference_type	2460
5.454.2.3	iterator_category	2460
5.454.2.4	pointer	2460
5.454.2.5	reference	2461
5.454.2.6	value_type	2461

5.454.3 Constructor & Destructor Documentation	2461
5.454.3.1 front_insert_iterator	2461
5.454.4 Member Function Documentation	2461
5.454.4.1 operator*	2461
5.454.4.2 operator++	2461
5.454.4.3 operator++	2462
5.454.4.4 operator=	2462
5.455std::function< _Res(_ArgTypes...)> Class Template Reference	2462
5.455.1 Detailed Description	2464
5.455.2 Constructor & Destructor Documentation	2465
5.455.2.1 function	2465
5.455.2.2 function	2465
5.455.2.3 function	2465
5.455.2.4 function	2465
5.455.2.5 function	2466
5.455.3 Member Function Documentation	2466
5.455.3.1 operator bool	2466
5.455.3.2 operator()	2467
5.455.3.3 operator=	2467
5.455.3.4 operator=	2467
5.455.3.5 operator=	2468
5.455.3.6 operator=	2468
5.455.3.7 operator=	2469
5.455.3.8 swap	2469
5.455.3.9 target	2469
5.455.3.10target	2470
5.455.3.11target_type	2470
5.456std::future< _Res > Class Template Reference	2470
5.456.1 Detailed Description	2472
5.456.2 Constructor & Destructor Documentation	2472
5.456.2.1 future	2472

5.456.3 Member Function Documentation	2472
5.456.3.1 _M_get_result	2472
5.456.3.2 get	2473
5.457std::future< _Res & > Class Template Reference	2473
5.457.1 Detailed Description	2474
5.457.2 Constructor & Destructor Documentation	2475
5.457.2.1 future	2475
5.457.3 Member Function Documentation	2475
5.457.3.1 _M_get_result	2475
5.457.3.2 get	2475
5.458std::future< void > Class Template Reference	2475
5.458.1 Detailed Description	2477
5.458.2 Constructor & Destructor Documentation	2477
5.458.2.1 future	2477
5.458.3 Member Function Documentation	2477
5.458.3.1 _M_get_result	2477
5.458.3.2 get	2477
5.459std::future_error Class Reference	2478
5.459.1 Detailed Description	2478
5.459.2 Member Function Documentation	2479
5.459.2.1 what	2479
5.460std::gamma_distribution< _RealType > Class Template Reference	2479
5.460.1 Detailed Description	2480
5.460.2 Member Typedef Documentation	2481
5.460.2.1 result_type	2481
5.460.3 Constructor & Destructor Documentation	2481
5.460.3.1 gamma_distribution	2481
5.460.4 Member Function Documentation	2481
5.460.4.1 alpha	2481
5.460.4.2 beta	2481
5.460.4.3 max	2481

5.460.4.4 min	2482
5.460.4.5 operator()	2482
5.460.4.6 operator()	2482
5.460.4.7 param	2482
5.460.4.8 param	2483
5.460.4.9 reset	2483
5.460.5 Friends And Related Function Documentation	2483
5.460.5.1 operator<<	2483
5.460.5.2 operator==	2484
5.460.5.3 operator>>	2484
5.461 std::gamma_distribution< _RealType >::param_type Struct Reference	2484
5.461.1 Detailed Description	2485
5.462 std::geometric_distribution< _IntType > Class Template Reference	2485
5.462.1 Detailed Description	2486
5.462.2 Member Typedef Documentation	2486
5.462.2.1 result_type	2486
5.462.3 Member Function Documentation	2486
5.462.3.1 max	2486
5.462.3.2 min	2487
5.462.3.3 operator()	2487
5.462.3.4 p	2487
5.462.3.5 param	2487
5.462.3.6 param	2487
5.462.3.7 reset	2488
5.463 std::geometric_distribution< _IntType >::param_type Struct Reference	2488
5.463.1 Detailed Description	2488
5.464 std::greater< _Tp > Struct Template Reference	2489
5.464.1 Detailed Description	2490
5.464.2 Member Typedef Documentation	2490
5.464.2.1 first_argument_type	2490
5.464.2.2 result_type	2490

5.464.2.3 second_argument_type	2490
5.465std::greater_equal< _Tp > Struct Template Reference	2490
5.465.1 Detailed Description	2491
5.465.2 Member Typedef Documentation	2492
5.465.2.1 first_argument_type	2492
5.465.2.2 result_type	2492
5.465.2.3 second_argument_type	2492
5.466std::gslice Class Reference	2492
5.466.1 Detailed Description	2493
5.467std::gslice_array< _Tp > Class Template Reference	2493
5.467.1 Detailed Description	2494
5.468std::has_nothrow_copy_assign< _Tp > Struct Template Reference . .	2495
5.468.1 Detailed Description	2495
5.469std::has_nothrow_copy_constructor< _Tp > Struct Template Reference	2495
5.469.1 Detailed Description	2495
5.470std::has_nothrow_default_constructor< _Tp > Struct Template Refer- ence	2496
5.470.1 Detailed Description	2496
5.471std::has_trivial_copy_assign< _Tp > Struct Template Reference . . .	2496
5.471.1 Detailed Description	2496
5.472std::has_trivial_copy_constructor< _Tp > Struct Template Reference	2497
5.472.1 Detailed Description	2497
5.473std::has_trivial_default_constructor< _Tp > Struct Template Reference	2497
5.473.1 Detailed Description	2497
5.474std::has_trivial_destructor< _Tp > Struct Template Reference	2497
5.474.1 Detailed Description	2498
5.475std::hash< _Tp > Struct Template Reference	2498
5.475.1 Detailed Description	2500
5.475.2 Member Typedef Documentation	2500
5.475.2.1 argument_type	2500
5.475.2.2 result_type	2500

5.476std::hash< __debug::bitset< _Nb > > Struct Template Reference . .	2500
5.476.1 Detailed Description	2501
5.476.2 Member Typedef Documentation	2502
5.476.2.1 argument_type	2502
5.476.2.2 result_type	2502
5.477std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference	2502
5.477.1 Detailed Description	2503
5.477.2 Member Typedef Documentation	2504
5.477.2.1 argument_type	2504
5.477.2.2 result_type	2504
5.478std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference	2504
5.478.1 Detailed Description	2505
5.478.2 Member Typedef Documentation	2506
5.478.2.1 argument_type	2506
5.478.2.2 result_type	2506
5.479std::hash< __gnu_cxx::throw_value_random > Struct Template Reference	2506
5.479.1 Detailed Description	2507
5.479.2 Member Typedef Documentation	2508
5.479.2.1 argument_type	2508
5.479.2.2 result_type	2508
5.480std::hash< __profile::bitset< _Nb > > Struct Template Reference . .	2508
5.480.1 Detailed Description	2509
5.480.2 Member Typedef Documentation	2510
5.480.2.1 argument_type	2510
5.480.2.2 result_type	2510
5.481std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference	2510
5.481.1 Detailed Description	2511
5.481.2 Member Typedef Documentation	2512
5.481.2.1 argument_type	2512

5.481.2.2 result_type	2512
5.482std::hash< _Tp * > Struct Template Reference	2512
5.482.1 Detailed Description	2513
5.482.2 Member Typedef Documentation	2514
5.482.2.1 argument_type	2514
5.482.2.2 result_type	2514
5.483std::hash< error_code > Struct Template Reference	2514
5.483.1 Detailed Description	2515
5.483.2 Member Typedef Documentation	2516
5.483.2.1 argument_type	2516
5.483.2.2 result_type	2516
5.484std::hash< shared_ptr< _Tp > > Struct Template Reference	2516
5.484.1 Detailed Description	2517
5.484.2 Member Typedef Documentation	2518
5.484.2.1 argument_type	2518
5.484.2.2 result_type	2518
5.485std::hash< string > Struct Template Reference	2518
5.485.1 Detailed Description	2519
5.485.2 Member Typedef Documentation	2520
5.485.2.1 argument_type	2520
5.485.2.2 result_type	2520
5.486std::hash< thread::id > Struct Template Reference	2520
5.486.1 Detailed Description	2521
5.486.2 Member Typedef Documentation	2522
5.486.2.1 argument_type	2522
5.486.2.2 result_type	2522
5.487std::hash< u16string > Struct Template Reference	2522
5.487.1 Detailed Description	2523
5.487.2 Member Typedef Documentation	2524
5.487.2.1 argument_type	2524
5.487.2.2 result_type	2524

5.488std::hash< u32string > Struct Template Reference	2524
5.488.1 Detailed Description	2525
5.488.2 Member Typedef Documentation	2526
5.488.2.1 argument_type	2526
5.488.2.2 result_type	2526
5.489std::hash< unique_ptr< _Tp, _Tp_Deleter > > Struct Template Reference	2526
5.489.1 Detailed Description	2527
5.489.2 Member Typedef Documentation	2528
5.489.2.1 argument_type	2528
5.489.2.2 result_type	2528
5.490std::hash< wstring > Struct Template Reference	2528
5.490.1 Detailed Description	2529
5.490.2 Member Typedef Documentation	2530
5.490.2.1 argument_type	2530
5.490.2.2 result_type	2530
5.491std::hash<::bitset< _Nb > > Struct Template Reference	2530
5.491.1 Detailed Description	2531
5.491.2 Member Typedef Documentation	2532
5.491.2.1 argument_type	2532
5.491.2.2 result_type	2532
5.492std::hash<::vector< bool, _Alloc > > Struct Template Reference	2532
5.492.1 Detailed Description	2533
5.492.2 Member Typedef Documentation	2534
5.492.2.1 argument_type	2534
5.492.2.2 result_type	2534
5.493std::identity< _Tp > Struct Template Reference	2534
5.493.1 Detailed Description	2534
5.494std::independent_bits_engine< _RandomNumberEngine, __w, _- UIntType > Class Template Reference	2535
5.494.1 Detailed Description	2536
5.494.2 Member Typedef Documentation	2536

5.494.2.1 result_type	2536
5.494.3 Constructor & Destructor Documentation	2536
5.494.3.1 independent_bits_engine	2536
5.494.3.2 independent_bits_engine	2536
5.494.3.3 independent_bits_engine	2537
5.494.3.4 independent_bits_engine	2537
5.494.3.5 independent_bits_engine	2537
5.494.4 Member Function Documentation	2538
5.494.4.1 base	2538
5.494.4.2 discard	2538
5.494.4.3 max	2538
5.494.4.4 min	2538
5.494.4.5 operator()	2539
5.494.4.6 seed	2539
5.494.4.7 seed	2539
5.494.4.8 seed	2539
5.494.5 Friends And Related Function Documentation	2540
5.494.5.1 operator==	2540
5.494.5.2 operator>>	2540
5.495std::indirect_array< _Tp > Class Template Reference	2541
5.495.1 Detailed Description	2542
5.495.2 Member Function Documentation	2542
5.495.2.1 operator%=	2542
5.495.2.2 operator&=	2543
5.495.2.3 operator*=	2543
5.495.2.4 operator+=	2543
5.495.2.5 operator-=	2543
5.495.2.6 operator/=	2543
5.495.2.7 operator<<=	2543
5.495.2.8 operator>>=	2543
5.495.2.9 operator^=	2543

5.495.2.10operator =	2544
5.496std::initializer_list<_E> Class Template Reference	2544
5.496.1 Detailed Description	2544
5.497std::input_iterator_tag Struct Reference	2545
5.497.1 Detailed Description	2545
5.498std::insert_iterator<_Container> Class Template Reference	2546
5.498.1 Detailed Description	2547
5.498.2 Member Typedef Documentation	2547
5.498.2.1 container_type	2547
5.498.2.2 difference_type	2547
5.498.2.3 iterator_category	2547
5.498.2.4 pointer	2548
5.498.2.5 reference	2548
5.498.2.6 value_type	2548
5.498.3 Constructor & Destructor Documentation	2548
5.498.3.1 insert_iterator	2548
5.498.4 Member Function Documentation	2548
5.498.4.1 operator*	2548
5.498.4.2 operator++	2549
5.498.4.3 operator++	2549
5.498.4.4 operator=	2549
5.499std::invalid_argument Class Reference	2550
5.499.1 Detailed Description	2550
5.499.2 Member Function Documentation	2550
5.499.2.1 what	2550
5.500std::ios_base Class Reference	2551
5.500.1 Detailed Description	2554
5.500.2 Member Typedef Documentation	2554
5.500.2.1 event_callback	2554
5.500.2.2 fmtflags	2555
5.500.2.3 iostate	2556

5.500.2.4 openmode	2556
5.500.2.5 seekdir	2556
5.500.3 Member Enumeration Documentation	2557
5.500.3.1 event	2557
5.500.4 Constructor & Destructor Documentation	2557
5.500.4.1 ~ios_base	2557
5.500.5 Member Function Documentation	2557
5.500.5.1 _M_getloc	2557
5.500.5.2 flags	2558
5.500.5.3 flags	2558
5.500.5.4 getloc	2558
5.500.5.5 imbue	2559
5.500.5.6 iword	2559
5.500.5.7 precision	2559
5.500.5.8 precision	2560
5.500.5.9 pword	2560
5.500.5.10register_callback	2560
5.500.5.11setf	2561
5.500.5.12setf	2561
5.500.5.13sync_with_stdio	2562
5.500.5.14unsetf	2562
5.500.5.15width	2562
5.500.5.16width	2563
5.500.5.17xalloc	2563
5.500.6 Member Data Documentation	2563
5.500.6.1 adjustfield	2563
5.500.6.2 app	2563
5.500.6.3 ate	2564
5.500.6.4 badbit	2564
5.500.6.5 basefield	2564
5.500.6.6 beg	2564

5.500.6.7 binary	2564
5.500.6.8 boolalpha	2565
5.500.6.9 cur	2565
5.500.6.10dec	2565
5.500.6.11end	2565
5.500.6.12eofbit	2565
5.500.6.13failbit	2566
5.500.6.14fixed	2566
5.500.6.15floatfield	2566
5.500.6.16goodbit	2566
5.500.6.17hex	2567
5.500.6.18n	2567
5.500.6.19internal	2567
5.500.6.20left	2567
5.500.6.21loct	2568
5.500.6.22out	2568
5.500.6.23right	2568
5.500.6.24scientific	2568
5.500.6.25showbase	2568
5.500.6.26showpoint	2568
5.500.6.27showpos	2568
5.500.6.28skipws	2569
5.500.6.29trunc	2569
5.500.6.30unitbuf	2569
5.500.6.31uppercase	2569
5.501std::ios_base::failure Class Reference	2569
5.501.1 Detailed Description	2570
5.501.2 Member Function Documentation	2570
5.501.2.1 what	2570
5.502std::is_base_of<_Base, _Derived> Struct Template Reference . . .	2571
5.502.1 Detailed Description	2571

5.503	<code>std::is_bind_expression< _Tp ></code> Struct Template Reference	2571
5.503.1	Detailed Description	2571
5.504	<code>std::is_bind_expression< _Bind< _Signature > ></code> Struct Template Reference	2572
5.504.1	Detailed Description	2572
5.505	<code>std::is_bind_expression< _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2572
5.505.1	Detailed Description	2572
5.506	<code>std::is_constructible< _Tp, _Args ></code> Struct Template Reference	2573
5.506.1	Detailed Description	2573
5.507	<code>std::is_convertible< _From, _To ></code> Struct Template Reference	2574
5.507.1	Detailed Description	2574
5.508	<code>std::is_error_code_enum< _Tp ></code> Struct Template Reference	2574
5.508.1	Detailed Description	2574
5.509	<code>std::is_error_condition_enum< _Tp ></code> Struct Template Reference	2575
5.509.1	Detailed Description	2575
5.510	<code>std::is_explicitly_convertible< _From, _To ></code> Struct Template Reference	2575
5.510.1	Detailed Description	2576
5.511	<code>std::is_lvalue_reference< typename ></code> Struct Template Reference	2576
5.511.1	Detailed Description	2577
5.512	<code>std::is_nothrow_constructible< _Tp, _Args ></code> Struct Template Reference	2577
5.512.1	Detailed Description	2577
5.513	<code>std::is_placeholder< _Tp ></code> Struct Template Reference	2577
5.513.1	Detailed Description	2578
5.514	<code>std::is_placeholder< _Placeholder< _Num > ></code> Struct Template Reference	2578
5.514.1	Detailed Description	2578
5.515	<code>std::is_pod< _Tp ></code> Struct Template Reference	2578
5.515.1	Detailed Description	2578
5.516	<code>std::is_reference< _Tp ></code> Struct Template Reference	2579
5.516.1	Detailed Description	2579
5.517	<code>std::is_rvalue_reference< typename ></code> Struct Template Reference	2579

5.517.1 Detailed Description	2579
5.518std::is_signed< _Tp > Struct Template Reference	2580
5.518.1 Detailed Description	2580
5.519std::is_standard_layout< _Tp > Struct Template Reference	2580
5.519.1 Detailed Description	2580
5.520std::is_trivial< _Tp > Struct Template Reference	2580
5.520.1 Detailed Description	2581
5.521std::is_unsigned< _Tp > Struct Template Reference	2581
5.521.1 Detailed Description	2581
5.522std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference	2581
5.522.1 Detailed Description	2583
5.522.2 Member Typedef Documentation	2583
5.522.2.1 difference_type	2583
5.522.2.2 iterator_category	2583
5.522.2.3 pointer	2583
5.522.2.4 reference	2583
5.522.2.5 value_type	2583
5.522.3 Constructor & Destructor Documentation	2584
5.522.3.1 istream_iterator	2584
5.522.3.2 istream_iterator	2584
5.523std::istreambuf_iterator< _CharT, _Traits > Class Template Reference	2584
5.523.1 Detailed Description	2585
5.523.2 Member Typedef Documentation	2586
5.523.2.1 char_type	2586
5.523.2.2 difference_type	2586
5.523.2.3 int_type	2586
5.523.2.4 istream_type	2586
5.523.2.5 iterator_category	2586
5.523.2.6 pointer	2586
5.523.2.7 reference	2587

5.523.2.8 streambuf_type	2587
5.523.2.9 traits_type	2587
5.523.2.10 value_type	2587
5.523.3 Constructor & Destructor Documentation	2587
5.523.3.1 istreambuf_iterator	2587
5.523.3.2 istreambuf_iterator	2588
5.523.3.3 istreambuf_iterator	2588
5.523.4 Member Function Documentation	2588
5.523.4.1 equal	2588
5.523.4.2 operator*	2588
5.523.4.3 operator++	2588
5.523.4.4 operator++	2589
5.524 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference	2589
5.524.1 Detailed Description	2590
5.524.2 Member Typedef Documentation	2590
5.524.2.1 difference_type	2590
5.524.2.2 iterator_category	2590
5.524.2.3 pointer	2590
5.524.2.4 reference	2591
5.524.2.5 value_type	2591
5.525 std::iterator_traits< _Iterator > Struct Template Reference	2591
5.525.1 Detailed Description	2592
5.526 std::iterator_traits< _Tp * > Struct Template Reference	2593
5.526.1 Detailed Description	2593
5.527 std::iterator_traits< const _Tp * > Struct Template Reference	2593
5.527.1 Detailed Description	2594
5.528 std::length_error Class Reference	2594
5.528.1 Detailed Description	2595
5.528.2 Member Function Documentation	2595
5.528.2.1 what	2595

5.529std::less< _Tp > Struct Template Reference	2595
5.529.1 Detailed Description	2596
5.529.2 Member Typedef Documentation	2597
5.529.2.1 first_argument_type	2597
5.529.2.2 result_type	2597
5.529.2.3 second_argument_type	2597
5.530std::less_equal< _Tp > Struct Template Reference	2597
5.530.1 Detailed Description	2598
5.530.2 Member Typedef Documentation	2599
5.530.2.1 first_argument_type	2599
5.530.2.2 result_type	2599
5.530.2.3 second_argument_type	2599
5.531std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	2599
5.531.1 Detailed Description	2600
5.531.2 Member Typedef Documentation	2601
5.531.2.1 result_type	2601
5.531.3 Constructor & Destructor Documentation	2601
5.531.3.1 linear_congruential_engine	2601
5.531.3.2 linear_congruential_engine	2601
5.531.4 Member Function Documentation	2602
5.531.4.1 discard	2602
5.531.4.2 max	2602
5.531.4.3 min	2602
5.531.4.4 operator()	2603
5.531.4.5 seed	2603
5.531.4.6 seed	2603
5.531.5 Friends And Related Function Documentation	2604
5.531.5.1 operator<<	2604
5.531.5.2 operator==	2604
5.531.5.3 operator>>	2605

5.531.6 Member Data Documentation	2605
5.531.6.1 increment	2605
5.531.6.2 modulus	2605
5.531.6.3 multiplier	2605
5.532std::list< _Tp, _Alloc > Class Template Reference	2606
5.532.1 Detailed Description	2610
5.532.2 Constructor & Destructor Documentation	2610
5.532.2.1 list	2610
5.532.2.2 list	2611
5.532.2.3 list	2611
5.532.2.4 list	2611
5.532.2.5 list	2612
5.532.2.6 list	2612
5.532.2.7 list	2612
5.532.2.8 list	2613
5.532.3 Member Function Documentation	2613
5.532.3.1 _M_create_node	2613
5.532.3.2 assign	2613
5.532.3.3 assign	2614
5.532.3.4 assign	2614
5.532.3.5 back	2614
5.532.3.6 back	2615
5.532.3.7 begin	2615
5.532.3.8 begin	2615
5.532.3.9 cbegin	2615
5.532.3.10end	2615
5.532.3.11clear	2616
5.532.3.12rbegin	2616
5.532.3.13rend	2616
5.532.3.14emplace	2616
5.532.3.15empty	2617

5.532.3.16end	2617
5.532.3.17end	2617
5.532.3.18erase	2617
5.532.3.19erase	2618
5.532.3.20front	2618
5.532.3.21front	2619
5.532.3.22get_allocator	2619
5.532.3.23insert	2619
5.532.3.24insert	2619
5.532.3.25insert	2620
5.532.3.26insert	2620
5.532.3.27insert	2621
5.532.3.28max_size	2621
5.532.3.29merge	2621
5.532.3.30merge	2622
5.532.3.31operator=	2622
5.532.3.32operator=	2623
5.532.3.33operator=	2623
5.532.3.34pop_back	2623
5.532.3.35pop_front	2623
5.532.3.36push_back	2624
5.532.3.37push_front	2624
5.532.3.38rbegin	2624
5.532.3.39rbegin	2625
5.532.3.40remove	2625
5.532.3.41remove_if	2625
5.532.3.42rend	2625
5.532.3.43rend	2626
5.532.3.44resize	2626
5.532.3.45resize	2626
5.532.3.46reverse	2627

5.532.3.47size	2627
5.532.3.48sort	2627
5.532.3.49sort	2627
5.532.3.50splice	2628
5.532.3.51splice	2628
5.532.3.52splice	2628
5.532.3.53swap	2629
5.532.3.54unique	2629
5.532.3.55unique	2630
5.533std::locale Class Reference	2630
5.533.1 Detailed Description	2632
5.533.2 Member Typedef Documentation	2632
5.533.2.1 category	2632
5.533.3 Constructor & Destructor Documentation	2632
5.533.3.1 locale	2632
5.533.3.2 locale	2632
5.533.3.3 locale	2633
5.533.3.4 locale	2633
5.533.3.5 locale	2633
5.533.3.6 locale	2634
5.533.3.7 ~locale	2634
5.533.4 Member Function Documentation	2634
5.533.4.1 classic	2634
5.533.4.2 combine	2634
5.533.4.3 global	2635
5.533.4.4 name	2635
5.533.4.5 operator!=	2635
5.533.4.6 operator()	2636
5.533.4.7 operator=	2636
5.533.4.8 operator==	2636
5.533.5 Friends And Related Function Documentation	2637

5.533.5.1 has_facet	2637
5.533.5.2 use_facet	2637
5.533.6 Member Data Documentation	2638
5.533.6.1 all	2638
5.533.6.2 collate	2638
5.533.6.3 ctype	2638
5.533.6.4 messages	2638
5.533.6.5 monetary	2639
5.533.6.6 none	2639
5.533.6.7 numeric	2639
5.533.6.8 time	2639
5.534std::locale::facet Class Reference	2640
5.534.1 Detailed Description	2641
5.534.2 Constructor & Destructor Documentation	2641
5.534.2.1 facet	2641
5.534.2.2 ~facet	2641
5.535std::locale::id Class Reference	2642
5.535.1 Detailed Description	2642
5.535.2 Constructor & Destructor Documentation	2642
5.535.2.1 id	2642
5.535.3 Friends And Related Function Documentation	2643
5.535.3.1 has_facet	2643
5.535.3.2 use_facet	2643
5.536std::lock_guard< _Mutex > Class Template Reference	2644
5.536.1 Detailed Description	2644
5.537std::logic_error Class Reference	2644
5.537.1 Detailed Description	2645
5.537.2 Constructor & Destructor Documentation	2645
5.537.2.1 logic_error	2645
5.537.3 Member Function Documentation	2646
5.537.3.1 what	2646

5.538std::logical_and< _Tp > Struct Template Reference	2646
5.538.1 Detailed Description	2647
5.538.2 Member Typedef Documentation	2647
5.538.2.1 first_argument_type	2647
5.538.2.2 result_type	2647
5.538.2.3 second_argument_type	2647
5.539std::logical_not< _Tp > Struct Template Reference	2648
5.539.1 Detailed Description	2648
5.539.2 Member Typedef Documentation	2649
5.539.2.1 argument_type	2649
5.539.2.2 result_type	2649
5.540std::logical_or< _Tp > Struct Template Reference	2649
5.540.1 Detailed Description	2650
5.540.2 Member Typedef Documentation	2651
5.540.2.1 first_argument_type	2651
5.540.2.2 result_type	2651
5.540.2.3 second_argument_type	2651
5.541std::lognormal_distribution< _RealType > Class Template Reference	2651
5.541.1 Detailed Description	2652
5.541.2 Member Typedef Documentation	2653
5.541.2.1 result_type	2653
5.541.3 Member Function Documentation	2653
5.541.3.1 max	2653
5.541.3.2 min	2653
5.541.3.3 operator()	2653
5.541.3.4 param	2653
5.541.3.5 param	2654
5.541.3.6 reset	2654
5.541.4 Friends And Related Function Documentation	2654
5.541.4.1 operator<<	2654
5.541.4.2 operator==	2655

5.541.4.3 operator>>	2655
5.542std::lognormal_distribution<_RealType>::param_type Struct Reference	2655
5.542.1 Detailed Description	2656
5.543std::make_signed<_Tp> Struct Template Reference	2656
5.543.1 Detailed Description	2656
5.544std::make_unsigned<_Tp> Struct Template Reference	2656
5.544.1 Detailed Description	2657
5.545std::map<_Key, _Tp, _Compare, _Alloc> Class Template Reference	2657
5.545.1 Detailed Description	2659
5.545.2 Constructor & Destructor Documentation	2660
5.545.2.1 map	2660
5.545.2.2 map	2660
5.545.2.3 map	2660
5.545.2.4 map	2661
5.545.2.5 map	2661
5.545.2.6 map	2661
5.545.2.7 map	2662
5.545.3 Member Function Documentation	2662
5.545.3.1 at	2662
5.545.3.2 begin	2663
5.545.3.3 begin	2663
5.545.3.4 cbegin	2663
5.545.3.5 cend	2663
5.545.3.6 clear	2664
5.545.3.7 count	2664
5.545.3.8 crbegin	2664
5.545.3.9 crend	2665
5.545.3.10empty	2665
5.545.3.11lend	2665
5.545.3.12end	2665

5.545.3.13	<code>equal_range</code>	2666
5.545.3.14	<code>equal_range</code>	2666
5.545.3.15	<code>erase</code>	2667
5.545.3.16	<code>erase</code>	2667
5.545.3.17	<code>erase</code>	2668
5.545.3.18	<code>find</code>	2668
5.545.3.19	<code>find</code>	2669
5.545.3.20	<code>get_allocator</code>	2669
5.545.3.21	<code>insert</code>	2669
5.545.3.22	<code>insert</code>	2670
5.545.3.23	<code>insert</code>	2670
5.545.3.24	<code>insert</code>	2671
5.545.3.25	<code>key_comp</code>	2671
5.545.3.26	<code>lower_bound</code>	2671
5.545.3.27	<code>lower_bound</code>	2672
5.545.3.28	<code>max_size</code>	2672
5.545.3.29	<code>operator=</code>	2672
5.545.3.30	<code>operator=</code>	2673
5.545.3.31	<code>operator=</code>	2673
5.545.3.32	<code>operator[]</code>	2674
5.545.3.33	<code>begin</code>	2674
5.545.3.34	<code>begin</code>	2674
5.545.3.35	<code>end</code>	2675
5.545.3.36	<code>end</code>	2675
5.545.3.37	<code>size</code>	2675
5.545.3.38	<code>swap</code>	2675
5.545.3.39	<code>upper_bound</code>	2676
5.545.3.40	<code>upper_bound</code>	2676
5.545.3.41	<code>value_comp</code>	2676
5.546	<code>std::mask_array< _Tp ></code> Class Template Reference	2677
5.546.1	Detailed Description	2678

5.546.2 Member Function Documentation	2678
5.546.2.1 operator%=	2678
5.546.2.2 operator&=	2679
5.546.2.3 operator*=	2679
5.546.2.4 operator+=	2679
5.546.2.5 operator-=	2679
5.546.2.6 operator/=	2679
5.546.2.7 operator<<=	2679
5.546.2.8 operator>>=	2679
5.546.2.9 operator^=	2679
5.546.2.10operator =	2680
5.547std::match_results< _Bi_iter, _Allocator > Class Template Reference	2680
5.547.1 Detailed Description	2684
5.547.2 Constructor & Destructor Documentation	2684
5.547.2.1 match_results	2684
5.547.2.2 match_results	2685
5.547.2.3 ~match_results	2685
5.547.3 Member Function Documentation	2685
5.547.3.1 begin	2685
5.547.3.2 cbegin	2685
5.547.3.3 cend	2686
5.547.3.4 empty	2686
5.547.3.5 end	2686
5.547.3.6 format	2687
5.547.3.7 format	2687
5.547.3.8 get_allocator	2687
5.547.3.9 length	2687
5.547.3.10max_size	2688
5.547.3.11operator=	2688
5.547.3.12operator[]	2688
5.547.3.13position	2689

5.547.3.14	prefix	2689
5.547.3.15	size	2689
5.547.3.16	str	2690
5.547.3.17	suffix	2690
5.547.3.18	swap	2690
5.548	std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2691
5.548.1	Detailed Description	2692
5.548.2	Member Typedef Documentation	2692
5.548.2.1	first_argument_type	2692
5.548.2.2	result_type	2692
5.548.2.3	second_argument_type	2692
5.549	std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2692
5.549.1	Detailed Description	2693
5.549.2	Member Typedef Documentation	2694
5.549.2.1	first_argument_type	2694
5.549.2.2	result_type	2694
5.549.2.3	second_argument_type	2694
5.550	std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2694
5.550.1	Detailed Description	2695
5.550.2	Member Typedef Documentation	2696
5.550.2.1	argument_type	2696
5.550.2.2	result_type	2696
5.551	std::mem_fun_t< _Ret, _Tp > Class Template Reference	2696
5.551.1	Detailed Description	2697
5.551.2	Member Typedef Documentation	2698
5.551.2.1	argument_type	2698
5.551.2.2	result_type	2698
5.552	std::messages< _CharT > Class Template Reference	2698
5.552.1	Detailed Description	2700
5.552.2	Member Typedef Documentation	2701
5.552.2.1	char_type	2701

5.552.2.2 string_type	2701
5.552.3 Constructor & Destructor Documentation	2701
5.552.3.1 messages	2701
5.552.3.2 messages	2702
5.552.3.3 ~messages	2702
5.552.4 Member Data Documentation	2702
5.552.4.1 id	2702
5.553std::messages_base Struct Reference	2702
5.553.1 Detailed Description	2703
5.554std::messages_byname< _CharT > Class Template Reference	2703
5.554.1 Detailed Description	2705
5.554.2 Member Typedef Documentation	2706
5.554.2.1 char_type	2706
5.554.2.2 string_type	2706
5.554.3 Member Data Documentation	2706
5.554.3.1 id	2706
5.555std::minus< _Tp > Struct Template Reference	2706
5.555.1 Detailed Description	2707
5.555.2 Member Typedef Documentation	2708
5.555.2.1 first_argument_type	2708
5.555.2.2 result_type	2708
5.555.2.3 second_argument_type	2708
5.556std::modulus< _Tp > Struct Template Reference	2708
5.556.1 Detailed Description	2709
5.556.2 Member Typedef Documentation	2710
5.556.2.1 first_argument_type	2710
5.556.2.2 result_type	2710
5.556.2.3 second_argument_type	2710
5.557std::money_base Class Reference	2710
5.557.1 Detailed Description	2711
5.558std::money_get< _CharT, _InIter > Class Template Reference	2712

5.558.1 Detailed Description	2714
5.558.2 Member Typedef Documentation	2714
5.558.2.1 char_type	2714
5.558.2.2 iter_type	2714
5.558.2.3 string_type	2714
5.558.3 Constructor & Destructor Documentation	2715
5.558.3.1 money_get	2715
5.558.3.2 ~money_get	2715
5.558.4 Member Function Documentation	2715
5.558.4.1 do_get	2715
5.558.4.2 do_get	2716
5.558.4.3 get	2716
5.558.4.4 get	2717
5.558.5 Member Data Documentation	2717
5.558.5.1 id	2717
5.559std::money_put< _CharT, _OutIter > Class Template Reference . . .	2718
5.559.1 Detailed Description	2719
5.559.2 Member Typedef Documentation	2720
5.559.2.1 char_type	2720
5.559.2.2 iter_type	2720
5.559.2.3 string_type	2720
5.559.3 Constructor & Destructor Documentation	2720
5.559.3.1 money_put	2720
5.559.3.2 ~money_put	2720
5.559.4 Member Function Documentation	2721
5.559.4.1 do_put	2721
5.559.4.2 do_put	2721
5.559.4.3 put	2722
5.559.4.4 put	2723
5.559.5 Member Data Documentation	2723
5.559.5.1 id	2723

5.560std::moneypunct< _CharT, _Intl > Class Template Reference	2724
5.560.1 Detailed Description	2726
5.560.2 Member Typedef Documentation	2726
5.560.2.1 char_type	2726
5.560.2.2 string_type	2727
5.560.3 Constructor & Destructor Documentation	2727
5.560.3.1 moneypunct	2727
5.560.3.2 moneypunct	2727
5.560.3.3 moneypunct	2727
5.560.3.4 ~moneypunct	2728
5.560.4 Member Function Documentation	2728
5.560.4.1 curr_symbol	2728
5.560.4.2 decimal_point	2728
5.560.4.3 do_curr_symbol	2729
5.560.4.4 do_decimal_point	2729
5.560.4.5 do_frac_digits	2729
5.560.4.6 do_grouping	2730
5.560.4.7 do_neg_format	2730
5.560.4.8 do_negative_sign	2731
5.560.4.9 do_pos_format	2731
5.560.4.10do_positive_sign	2731
5.560.4.11do_thousands_sep	2732
5.560.4.12frac_digits	2732
5.560.4.13grouping	2733
5.560.4.14neg_format	2733
5.560.4.15negative_sign	2734
5.560.4.16pos_format	2734
5.560.4.17positive_sign	2735
5.560.4.18thousands_sep	2735
5.560.5 Member Data Documentation	2736
5.560.5.1 id	2736

5.560.5.2 intl	2736
5.561 std::moneypunct_byname< _CharT, _Intl > Class Template Reference	2736
5.561.1 Detailed Description	2739
5.561.2 Member Typedef Documentation	2739
5.561.2.1 char_type	2739
5.561.2.2 string_type	2739
5.561.3 Member Function Documentation	2740
5.561.3.1 curr_symbol	2740
5.561.3.2 decimal_point	2740
5.561.3.3 do_curr_symbol	2740
5.561.3.4 do_decimal_point	2741
5.561.3.5 do_frac_digits	2741
5.561.3.6 do_grouping	2742
5.561.3.7 do_neg_format	2742
5.561.3.8 do_negative_sign	2742
5.561.3.9 do_pos_format	2743
5.561.3.10 do_positive_sign	2743
5.561.3.11 do_thousands_sep	2744
5.561.3.12 frac_digits	2744
5.561.3.13 grouping	2744
5.561.3.14 neg_format	2745
5.561.3.15 negative_sign	2746
5.561.3.16 pos_format	2746
5.561.3.17 positive_sign	2747
5.561.3.18 thousands_sep	2747
5.561.4 Member Data Documentation	2748
5.561.4.1 id	2748
5.561.4.2 intl	2748
5.562 std::move_iterator< _Iterator > Class Template Reference	2748
5.562.1 Detailed Description	2749

5.563 <code>std::multimap< _Key, _Tp, _Compare, _Alloc ></code> Class Template Reference	2749
5.563.1 Detailed Description	2752
5.563.2 Constructor & Destructor Documentation	2752
5.563.2.1 <code>multimap</code>	2752
5.563.2.2 <code>multimap</code>	2752
5.563.2.3 <code>multimap</code>	2753
5.563.2.4 <code>multimap</code>	2753
5.563.2.5 <code>multimap</code>	2753
5.563.2.6 <code>multimap</code>	2754
5.563.2.7 <code>multimap</code>	2754
5.563.3 Member Function Documentation	2755
5.563.3.1 <code>begin</code>	2755
5.563.3.2 <code>begin</code>	2755
5.563.3.3 <code>cbegin</code>	2755
5.563.3.4 <code>cend</code>	2755
5.563.3.5 <code>clear</code>	2756
5.563.3.6 <code>count</code>	2756
5.563.3.7 <code>crbegin</code>	2756
5.563.3.8 <code>crend</code>	2756
5.563.3.9 <code>empty</code>	2757
5.563.3.10 <code>end</code>	2757
5.563.3.11 <code>lend</code>	2757
5.563.3.12 <code>equal_range</code>	2757
5.563.3.13 <code>equal_range</code>	2758
5.563.3.14 <code>erase</code>	2758
5.563.3.15 <code>erase</code>	2759
5.563.3.16 <code>erase</code>	2759
5.563.3.17 <code>find</code>	2760
5.563.3.18 <code>find</code>	2760
5.563.3.19 <code>get_allocator</code>	2761

5.563.3.20	insert	2761
5.563.3.21	insert	2761
5.563.3.22	insert	2762
5.563.3.23	insert	2762
5.563.3.24	key_comp	2763
5.563.3.25	lower_bound	2763
5.563.3.26	lower_bound	2763
5.563.3.27	max_size	2764
5.563.3.28	operator=	2764
5.563.3.29	operator=	2764
5.563.3.30	operator=	2765
5.563.3.31	lrbegin	2765
5.563.3.32	begin	2765
5.563.3.33	rend	2766
5.563.3.34	rend	2766
5.563.3.35	size	2766
5.563.3.36	swap	2766
5.563.3.37	upper_bound	2767
5.563.3.38	upper_bound	2767
5.563.3.39	value_comp	2767
5.564	std::multiplies< _Tp > Struct Template Reference	2768
5.564.1	Detailed Description	2769
5.564.2	Member Typedef Documentation	2769
5.564.2.1	first_argument_type	2769
5.564.2.2	result_type	2769
5.564.2.3	second_argument_type	2769
5.565	std::multiset< _Key, _Compare, _Alloc > Class Template Reference	2769
5.565.1	Detailed Description	2771
5.565.2	Constructor & Destructor Documentation	2772
5.565.2.1	multiset	2772
5.565.2.2	multiset	2772

5.565.2.3 multiset	2772
5.565.2.4 multiset	2773
5.565.2.5 multiset	2773
5.565.2.6 multiset	2774
5.565.2.7 multiset	2774
5.565.3 Member Function Documentation	2774
5.565.3.1 begin	2774
5.565.3.2 cbegin	2775
5.565.3.3 cend	2775
5.565.3.4 clear	2775
5.565.3.5 count	2775
5.565.3.6 cbegin	2776
5.565.3.7 crend	2776
5.565.3.8 empty	2776
5.565.3.9 end	2776
5.565.3.10equal_range	2776
5.565.3.11equal_range	2777
5.565.3.12erase	2777
5.565.3.13erase	2778
5.565.3.14erase	2778
5.565.3.15find	2779
5.565.3.16find	2779
5.565.3.17get_allocator	2780
5.565.3.18insert	2780
5.565.3.19insert	2780
5.565.3.20insert	2781
5.565.3.21insert	2781
5.565.3.22key_comp	2782
5.565.3.23lower_bound	2782
5.565.3.24lower_bound	2782
5.565.3.25max_size	2783

5.565.3.26operator=	2783
5.565.3.27operator=	2783
5.565.3.28operator=	2784
5.565.3.29begin	2784
5.565.3.30end	2784
5.565.3.31size	2784
5.565.3.32swap	2785
5.565.3.33upper_bound	2785
5.565.3.34upper_bound	2785
5.565.3.35value_comp	2786
5.565.4 Friends And Related Function Documentation	2786
5.565.4.1 operator<	2786
5.565.4.2 operator==	2787
5.566std::mutex Class Reference	2787
5.566.1 Detailed Description	2788
5.567std::negate< _Tp > Struct Template Reference	2788
5.567.1 Detailed Description	2789
5.567.2 Member Typedef Documentation	2789
5.567.2.1 argument_type	2789
5.567.2.2 result_type	2789
5.568std::negative_binomial_distribution< _IntType > Class Template Reference	2789
5.568.1 Detailed Description	2790
5.568.2 Member Typedef Documentation	2791
5.568.2.1 result_type	2791
5.568.3 Member Function Documentation	2791
5.568.3.1 k	2791
5.568.3.2 max	2791
5.568.3.3 min	2791
5.568.3.4 operator()	2791
5.568.3.5 p	2792

5.568.3.6 param	2792
5.568.3.7 param	2792
5.568.3.8 reset	2792
5.568.4 Friends And Related Function Documentation	2793
5.568.4.1 operator<<	2793
5.568.4.2 operator==	2793
5.568.4.3 operator>>	2793
5.569std::negative_binomial_distribution< _IntType >::param_type Struct Reference	2794
5.569.1 Detailed Description	2794
5.570std::nested_exception Class Reference	2794
5.570.1 Detailed Description	2795
5.571std::normal_distribution< _RealType > Class Template Reference	2795
5.571.1 Detailed Description	2796
5.571.2 Member Typedef Documentation	2797
5.571.2.1 result_type	2797
5.571.3 Constructor & Destructor Documentation	2797
5.571.3.1 normal_distribution	2797
5.571.4 Member Function Documentation	2797
5.571.4.1 max	2797
5.571.4.2 mean	2797
5.571.4.3 min	2798
5.571.4.4 operator()	2798
5.571.4.5 operator()	2798
5.571.4.6 param	2798
5.571.4.7 param	2799
5.571.4.8 reset	2799
5.571.4.9 stddev	2799
5.571.5 Friends And Related Function Documentation	2799
5.571.5.1 operator<<	2799
5.571.5.2 operator==	2800

5.571.5.3 operator>>	2800
5.572std::normal_distribution< _RealType >::param_type Struct Reference	2800
5.572.1 Detailed Description	2801
5.573std::not_equal_to< _Tp > Struct Template Reference	2801
5.573.1 Detailed Description	2802
5.573.2 Member Typedef Documentation	2803
5.573.2.1 first_argument_type	2803
5.573.2.2 result_type	2803
5.573.2.3 second_argument_type	2803
5.574std::num_get< _CharT, _InIter > Class Template Reference	2803
5.574.1 Detailed Description	2806
5.574.2 Member Typedef Documentation	2806
5.574.2.1 char_type	2806
5.574.2.2 iter_type	2807
5.574.3 Constructor & Destructor Documentation	2807
5.574.3.1 num_get	2807
5.574.3.2 ~num_get	2807
5.574.4 Member Function Documentation	2807
5.574.4.1 do_get	2807
5.574.4.2 do_get	2808
5.574.4.3 do_get	2809
5.574.4.4 do_get	2809
5.574.4.5 do_get	2810
5.574.4.6 do_get	2810
5.574.4.7 do_get	2811
5.574.4.8 do_get	2812
5.574.4.9 do_get	2812
5.574.4.10do_get	2813
5.574.4.11do_get	2814
5.574.4.12get	2814
5.574.4.13get	2815

5.574.4.14get	2816
5.574.4.15get	2817
5.574.4.16get	2817
5.574.4.17get	2818
5.574.4.18get	2819
5.574.4.19get	2820
5.574.4.20get	2821
5.574.4.21get	2821
5.574.4.22get	2822
5.574.5 Member Data Documentation	2823
5.574.5.1 id	2823
5.575std::num_put<_CharT, _OutIter > Class Template Reference	2823
5.575.1 Detailed Description	2826
5.575.2 Member Typedef Documentation	2826
5.575.2.1 char_type	2826
5.575.2.2 iter_type	2826
5.575.3 Constructor & Destructor Documentation	2827
5.575.3.1 num_put	2827
5.575.3.2 ~num_put	2827
5.575.4 Member Function Documentation	2827
5.575.4.1 do_put	2827
5.575.4.2 do_put	2828
5.575.4.3 do_put	2828
5.575.4.4 do_put	2829
5.575.4.5 do_put	2829
5.575.4.6 do_put	2830
5.575.4.7 do_put	2830
5.575.4.8 do_put	2831
5.575.4.9 put	2831
5.575.4.10put	2832
5.575.4.11put	2833

5.575.4.12put	2834
5.575.4.13put	2834
5.575.4.14put	2835
5.575.4.15put	2836
5.575.4.16put	2837
5.575.5 Member Data Documentation	2838
5.575.5.1 id	2838
5.576std::numeric_limits< _Tp > Struct Template Reference	2838
5.576.1 Detailed Description	2840
5.576.2 Member Function Documentation	2840
5.576.2.1 denorm_min	2840
5.576.2.2 epsilon	2840
5.576.2.3 infinity	2840
5.576.2.4 lowest	2840
5.576.2.5 max	2841
5.576.2.6 min	2841
5.576.2.7 quiet_NaN	2841
5.576.2.8 round_error	2841
5.576.2.9 signaling_NaN	2841
5.576.3 Member Data Documentation	2841
5.576.3.1 digits	2841
5.576.3.2 digits10	2842
5.576.3.3 has_denorm	2842
5.576.3.4 has_denorm_loss	2842
5.576.3.5 has_infinity	2842
5.576.3.6 has_quiet_NaN	2842
5.576.3.7 has_signaling_NaN	2842
5.576.3.8 is_bounded	2843
5.576.3.9 is_exact	2843
5.576.3.10is_iec559	2843
5.576.3.11is_integer	2843

5.576.3.12	<code>s_modulo</code>	2843
5.576.3.13	<code>s_signed</code>	2843
5.576.3.14	<code>s_specialized</code>	2844
5.576.3.15	<code>max_digits10</code>	2844
5.576.3.16	<code>max_exponent</code>	2844
5.576.3.17	<code>max_exponent10</code>	2844
5.576.3.18	<code>min_exponent</code>	2844
5.576.3.19	<code>min_exponent10</code>	2845
5.576.3.20	<code>radix</code>	2845
5.576.3.21	<code>round_style</code>	2845
5.576.3.22	<code>traps_before</code>	2845
5.576.3.23	<code>traps</code>	2845
5.577	<code>std::numeric_limits< bool > Struct Template Reference</code>	2846
5.577.1	Detailed Description	2847
5.578	<code>std::numeric_limits< char > Struct Template Reference</code>	2847
5.578.1	Detailed Description	2848
5.579	<code>std::numeric_limits< char16_t > Struct Template Reference</code>	2848
5.579.1	Detailed Description	2849
5.580	<code>std::numeric_limits< char32_t > Struct Template Reference</code>	2849
5.580.1	Detailed Description	2851
5.581	<code>std::numeric_limits< double > Struct Template Reference</code>	2851
5.581.1	Detailed Description	2852
5.582	<code>std::numeric_limits< float > Struct Template Reference</code>	2852
5.582.1	Detailed Description	2853
5.583	<code>std::numeric_limits< int > Struct Template Reference</code>	2853
5.583.1	Detailed Description	2855
5.584	<code>std::numeric_limits< long > Struct Template Reference</code>	2855
5.584.1	Detailed Description	2856
5.585	<code>std::numeric_limits< long double > Struct Template Reference</code>	2856
5.585.1	Detailed Description	2857
5.586	<code>std::numeric_limits< long long > Struct Template Reference</code>	2857

5.586.1 Detailed Description	2859
5.587std::numeric_limits< short > Struct Template Reference	2859
5.587.1 Detailed Description	2860
5.588std::numeric_limits< signed char > Struct Template Reference	2860
5.588.1 Detailed Description	2861
5.589std::numeric_limits< unsigned char > Struct Template Reference	2861
5.589.1 Detailed Description	2863
5.590std::numeric_limits< unsigned int > Struct Template Reference	2863
5.590.1 Detailed Description	2864
5.591std::numeric_limits< unsigned long > Struct Template Reference	2864
5.591.1 Detailed Description	2865
5.592std::numeric_limits< unsigned long long > Struct Template Reference	2865
5.592.1 Detailed Description	2867
5.593std::numeric_limits< unsigned short > Struct Template Reference	2867
5.593.1 Detailed Description	2868
5.594std::numeric_limits< wchar_t > Struct Template Reference	2868
5.594.1 Detailed Description	2869
5.595std::num_punct< _CharT > Class Template Reference	2869
5.595.1 Detailed Description	2872
5.595.2 Member Typedef Documentation	2872
5.595.2.1 char_type	2872
5.595.2.2 string_type	2872
5.595.3 Constructor & Destructor Documentation	2872
5.595.3.1 num_punct	2872
5.595.3.2 num_punct	2873
5.595.3.3 num_punct	2873
5.595.3.4 ~num_punct	2873
5.595.4 Member Function Documentation	2873
5.595.4.1 decimal_point	2873
5.595.4.2 do_decimal_point	2874
5.595.4.3 do_falsename	2874

5.595.4.4 do_grouping	2875
5.595.4.5 do_thousands_sep	2875
5.595.4.6 do_truename	2875
5.595.4.7 falsename	2876
5.595.4.8 grouping	2876
5.595.4.9 thousands_sep	2877
5.595.4.10truename	2877
5.595.5 Member Data Documentation	2877
5.595.5.1 id	2877
5.596std::num_punct_byname< _CharT > Class Template Reference	2878
5.596.1 Detailed Description	2879
5.596.2 Member Typedef Documentation	2880
5.596.2.1 char_type	2880
5.596.2.2 string_type	2880
5.596.3 Member Function Documentation	2880
5.596.3.1 decimal_point	2880
5.596.3.2 do_decimal_point	2880
5.596.3.3 do_falsename	2881
5.596.3.4 do_grouping	2881
5.596.3.5 do_thousands_sep	2882
5.596.3.6 do_truename	2882
5.596.3.7 falsename	2882
5.596.3.8 grouping	2883
5.596.3.9 thousands_sep	2883
5.596.3.10truename	2883
5.596.4 Member Data Documentation	2884
5.596.4.1 id	2884
5.597std::once_flag Struct Reference	2884
5.597.1 Detailed Description	2884
5.597.2 Friends And Related Function Documentation	2885
5.597.2.1 call_once	2885

5.598	std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	2885
5.598.1	Detailed Description	2886
5.598.2	Member Typedef Documentation	2886
5.598.2.1	char_type	2886
5.598.2.2	difference_type	2887
5.598.2.3	iterator_category	2887
5.598.2.4	ostream_type	2887
5.598.2.5	pointer	2887
5.598.2.6	reference	2887
5.598.2.7	traits_type	2887
5.598.2.8	value_type	2888
5.598.3	Constructor & Destructor Documentation	2888
5.598.3.1	ostream_iterator	2888
5.598.3.2	ostream_iterator	2888
5.598.3.3	ostream_iterator	2888
5.598.4	Member Function Documentation	2889
5.598.4.1	operator=	2889
5.599	std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	2889
5.599.1	Detailed Description	2890
5.599.2	Member Typedef Documentation	2891
5.599.2.1	char_type	2891
5.599.2.2	difference_type	2891
5.599.2.3	iterator_category	2891
5.599.2.4	ostream_type	2891
5.599.2.5	pointer	2891
5.599.2.6	reference	2891
5.599.2.7	streambuf_type	2892
5.599.2.8	traits_type	2892
5.599.2.9	value_type	2892
5.599.3	Constructor & Destructor Documentation	2892
5.599.3.1	ostreambuf_iterator	2892

5.599.3.2 ostreambuf_iterator	2892
5.599.4 Member Function Documentation	2893
5.599.4.1 failed	2893
5.599.4.2 operator*	2893
5.599.4.3 operator++	2893
5.599.4.4 operator++	2893
5.599.4.5 operator=	2893
5.600std::out_of_range Class Reference	2894
5.600.1 Detailed Description	2894
5.600.2 Member Function Documentation	2894
5.600.2.1 what	2894
5.601std::output_iterator_tag Struct Reference	2895
5.601.1 Detailed Description	2895
5.602std::overflow_error Class Reference	2896
5.602.1 Detailed Description	2896
5.602.2 Member Function Documentation	2896
5.602.2.1 what	2896
5.603std::owner_less< shared_ptr< _Tp > > Struct Template Reference	2897
5.603.1 Detailed Description	2897
5.604std::owner_less< weak_ptr< _Tp > > Struct Template Reference	2897
5.604.1 Detailed Description	2897
5.605std::packaged_task< _Res(_ArgTypes...) > Class Template Reference	2898
5.605.1 Detailed Description	2898
5.606std::pair< _T1, _T2 > Struct Template Reference	2899
5.606.1 Detailed Description	2900
5.606.2 Member Typedef Documentation	2900
5.606.2.1 first_type	2900
5.606.2.2 second_type	2900
5.606.3 Constructor & Destructor Documentation	2900
5.606.3.1 pair	2900
5.606.3.2 pair	2901

5.606.3.3 pair	2901
5.606.4 Member Data Documentation	2901
5.606.4.1 first	2901
5.606.4.2 second	2901
5.607std::piecewise_constant_distribution< _RealType > Class Template Reference	2901
5.607.1 Detailed Description	2903
5.607.2 Member Typedef Documentation	2903
5.607.2.1 result_type	2903
5.607.3 Member Function Documentation	2903
5.607.3.1 densities	2903
5.607.3.2 intervals	2903
5.607.3.3 max	2903
5.607.3.4 min	2904
5.607.3.5 operator()	2904
5.607.3.6 param	2904
5.607.3.7 param	2904
5.607.3.8 reset	2905
5.607.4 Friends And Related Function Documentation	2905
5.607.4.1 operator<<	2905
5.607.4.2 operator>>	2905
5.608std::piecewise_constant_distribution< _RealType >::param_type Struct Reference	2906
5.608.1 Detailed Description	2906
5.609std::piecewise_linear_distribution< _RealType > Class Template Ref- erence	2907
5.609.1 Detailed Description	2908
5.609.2 Member Typedef Documentation	2908
5.609.2.1 result_type	2908
5.609.3 Member Function Documentation	2908
5.609.3.1 densities	2908
5.609.3.2 intervals	2908

5.609.3.3 max	2909
5.609.3.4 min	2909
5.609.3.5 operator()	2909
5.609.3.6 param	2909
5.609.3.7 param	2910
5.609.3.8 reset	2910
5.609.4 Friends And Related Function Documentation	2910
5.609.4.1 operator<<	2910
5.609.4.2 operator>>	2911
5.610std::piecewise_linear_distribution< _RealType >::param_type Struct Reference	2911
5.610.1 Detailed Description	2912
5.611std::plus< _Tp > Struct Template Reference	2912
5.611.1 Detailed Description	2913
5.611.2 Member Typedef Documentation	2913
5.611.2.1 first_argument_type	2913
5.611.2.2 result_type	2913
5.611.2.3 second_argument_type	2913
5.612std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference	2914
5.612.1 Detailed Description	2915
5.612.2 Member Typedef Documentation	2915
5.612.2.1 first_argument_type	2915
5.612.2.2 result_type	2915
5.612.2.3 second_argument_type	2915
5.613std::pointer_to_unary_function< _Arg, _Result > Class Template Ref- erence	2915
5.613.1 Detailed Description	2916
5.613.2 Member Typedef Documentation	2917
5.613.2.1 argument_type	2917
5.613.2.2 result_type	2917
5.614std::poisson_distribution< _IntType > Class Template Reference . . .	2917

5.614.1 Detailed Description	2918
5.614.2 Member Typedef Documentation	2918
5.614.2.1 result_type	2918
5.614.3 Member Function Documentation	2919
5.614.3.1 max	2919
5.614.3.2 mean	2919
5.614.3.3 min	2919
5.614.3.4 operator()	2919
5.614.3.5 operator()	2920
5.614.3.6 param	2920
5.614.3.7 param	2920
5.614.3.8 reset	2920
5.614.4 Friends And Related Function Documentation	2921
5.614.4.1 operator<<	2921
5.614.4.2 operator==	2921
5.614.4.3 operator>>	2921
5.615std::poisson_distribution< _IntType >::param_type Struct Reference	2922
5.615.1 Detailed Description	2922
5.616std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	2922
5.616.1 Detailed Description	2923
5.616.2 Constructor & Destructor Documentation	2924
5.616.2.1 priority_queue	2924
5.616.2.2 priority_queue	2925
5.616.3 Member Function Documentation	2925
5.616.3.1 empty	2925
5.616.3.2 pop	2925
5.616.3.3 push	2926
5.616.3.4 size	2926
5.616.3.5 top	2926
5.617std::promise< _Res > Class Template Reference	2927

5.617.1 Detailed Description	2927
5.618std::promise< _Res & > Class Template Reference	2928
5.618.1 Detailed Description	2928
5.619std::promise< void > Class Template Reference	2928
5.619.1 Detailed Description	2929
5.620std::queue< _Tp, _Sequence > Class Template Reference	2929
5.620.1 Detailed Description	2930
5.620.2 Constructor & Destructor Documentation	2931
5.620.2.1 queue	2931
5.620.3 Member Function Documentation	2931
5.620.3.1 back	2931
5.620.3.2 back	2931
5.620.3.3 empty	2931
5.620.3.4 front	2931
5.620.3.5 front	2932
5.620.3.6 pop	2932
5.620.3.7 push	2932
5.620.3.8 size	2932
5.620.4 Member Data Documentation	2933
5.620.4.1 c	2933
5.621std::random_access_iterator_tag Struct Reference	2933
5.621.1 Detailed Description	2934
5.622std::random_device Class Reference	2934
5.622.1 Detailed Description	2935
5.622.2 Member Typedef Documentation	2935
5.622.2.1 result_type	2935
5.623std::range_error Class Reference	2936
5.623.1 Detailed Description	2936
5.623.2 Member Function Documentation	2936
5.623.2.1 what	2936
5.624std::ratio< _Num, _Den > Struct Template Reference	2937

5.624.1 Detailed Description	2937
5.625std::ratio_add< _R1, _R2 > Struct Template Reference	2938
5.625.1 Detailed Description	2938
5.626std::ratio_divide< _R1, _R2 > Struct Template Reference	2938
5.626.1 Detailed Description	2939
5.627std::ratio_equal< _R1, _R2 > Struct Template Reference	2939
5.627.1 Detailed Description	2939
5.628std::ratio_greater< _R1, _R2 > Struct Template Reference	2939
5.628.1 Detailed Description	2939
5.629std::ratio_greater_equal< _R1, _R2 > Struct Template Reference	2940
5.629.1 Detailed Description	2940
5.630std::ratio_less< _R1, _R2 > Struct Template Reference	2940
5.630.1 Detailed Description	2940
5.631std::ratio_less_equal< _R1, _R2 > Struct Template Reference	2941
5.631.1 Detailed Description	2941
5.632std::ratio_multiply< _R1, _R2 > Struct Template Reference	2941
5.632.1 Detailed Description	2941
5.633std::ratio_not_equal< _R1, _R2 > Struct Template Reference	2942
5.633.1 Detailed Description	2942
5.634std::ratio_subtract< _R1, _R2 > Struct Template Reference	2942
5.634.1 Detailed Description	2942
5.635std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	2943
5.635.1 Detailed Description	2944
5.635.2 Member Typedef Documentation	2944
5.635.2.1 difference_type	2944
5.635.2.2 iterator_category	2944
5.635.2.3 pointer	2944
5.635.2.4 reference	2944
5.635.2.5 value_type	2945
5.636std::recursive_mutex Class Reference	2945

5.636.1 Detailed Description	2945
5.637std::recursive_timed_mutex Class Reference	2945
5.637.1 Detailed Description	2946
5.638std::reference_wrapper< _Tp > Class Template Reference	2946
5.638.1 Detailed Description	2948
5.639std::regex_error Class Reference	2948
5.639.1 Detailed Description	2949
5.639.2 Constructor & Destructor Documentation	2949
5.639.2.1 regex_error	2949
5.639.3 Member Function Documentation	2949
5.639.3.1 code	2949
5.639.3.2 what	2949
5.640std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	2950
5.640.1 Detailed Description	2950
5.640.2 Constructor & Destructor Documentation	2951
5.640.2.1 regex_iterator	2951
5.640.2.2 regex_iterator	2951
5.640.2.3 regex_iterator	2951
5.640.3 Member Function Documentation	2952
5.640.3.1 operator!=	2952
5.640.3.2 operator*	2952
5.640.3.3 operator++	2952
5.640.3.4 operator++	2953
5.640.3.5 operator->	2953
5.640.3.6 operator=	2953
5.640.3.7 operator==	2953
5.641std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	2954
5.641.1 Detailed Description	2955
5.641.2 Constructor & Destructor Documentation	2955
5.641.2.1 regex_token_iterator	2955

5.641.2.2 <code>regex_token_iterator</code>	2955
5.641.2.3 <code>regex_token_iterator</code>	2956
5.641.2.4 <code>regex_token_iterator</code>	2957
5.641.2.5 <code>regex_token_iterator</code>	2957
5.641.3 Member Function Documentation	2958
5.641.3.1 <code>operator!=</code>	2958
5.641.3.2 <code>operator*</code>	2958
5.641.3.3 <code>operator++</code>	2958
5.641.3.4 <code>operator++</code>	2959
5.641.3.5 <code>operator-></code>	2959
5.641.3.6 <code>operator=</code>	2959
5.641.3.7 <code>operator==</code>	2960
5.642 <code>std::regex_traits<_Ch_type></code> Struct Template Reference	2960
5.642.1 Detailed Description	2961
5.642.2 Constructor & Destructor Documentation	2961
5.642.2.1 <code>regex_traits</code>	2961
5.642.3 Member Function Documentation	2961
5.642.3.1 <code>getloc</code>	2961
5.642.3.2 <code>imbue</code>	2962
5.642.3.3 <code>length</code>	2962
5.642.3.4 <code>lookup_classname</code>	2962
5.642.3.5 <code>lookup_collatename</code>	2964
5.642.3.6 <code>transform</code>	2964
5.642.3.7 <code>transform_primary</code>	2965
5.642.3.8 <code>translate</code>	2965
5.642.3.9 <code>translate_nocase</code>	2966
5.643 <code>std::remove_reference<_Tp></code> Struct Template Reference	2966
5.643.1 Detailed Description	2966
5.644 <code>std::reverse_iterator<_Iterator></code> Class Template Reference	2967
5.644.1 Detailed Description	2968
5.644.2 Member Typedef Documentation	2968

5.644.2.1 difference_type	2968
5.644.2.2 iterator_category	2968
5.644.2.3 pointer	2969
5.644.2.4 reference	2969
5.644.2.5 value_type	2969
5.644.3 Constructor & Destructor Documentation	2969
5.644.3.1 reverse_iterator	2969
5.644.3.2 reverse_iterator	2970
5.644.3.3 reverse_iterator	2970
5.644.3.4 reverse_iterator	2970
5.644.4 Member Function Documentation	2970
5.644.4.1 base	2970
5.644.4.2 operator*	2970
5.644.4.3 operator+	2971
5.644.4.4 operator++	2971
5.644.4.5 operator++	2971
5.644.4.6 operator+=	2972
5.644.4.7 operator-	2972
5.644.4.8 operator--	2972
5.644.4.9 operator--	2973
5.644.4.10operator-=	2973
5.644.4.11operator->	2973
5.644.4.12operator[]	2973
5.645std::runtime_error Class Reference	2974
5.645.1 Detailed Description	2975
5.645.2 Constructor & Destructor Documentation	2975
5.645.2.1 runtime_error	2975
5.645.3 Member Function Documentation	2975
5.645.3.1 what	2975
5.646std::seed_seq Class Reference	2975
5.646.1 Detailed Description	2976

5.646.2 Member Typedef Documentation	2976
5.646.2.1 result_type	2976
5.646.3 Constructor & Destructor Documentation	2976
5.646.3.1 seed_seq	2976
5.647 std::set< _Key, _Compare, _Alloc > Class Template Reference . . .	2976
5.647.1 Detailed Description	2979
5.647.2 Member Typedef Documentation	2979
5.647.2.1 allocator_type	2979
5.647.2.2 const_iterator	2980
5.647.2.3 const_pointer	2980
5.647.2.4 const_reference	2980
5.647.2.5 const_reverse_iterator	2980
5.647.2.6 difference_type	2980
5.647.2.7 iterator	2981
5.647.2.8 key_compare	2981
5.647.2.9 key_type	2981
5.647.2.10 pointer	2981
5.647.2.11 reference	2981
5.647.2.12 reverse_iterator	2982
5.647.2.13 size_type	2982
5.647.2.14 value_compare	2982
5.647.2.15 value_type	2982
5.647.3 Constructor & Destructor Documentation	2982
5.647.3.1 set	2982
5.647.3.2 set	2983
5.647.3.3 set	2983
5.647.3.4 set	2983
5.647.3.5 set	2984
5.647.3.6 set	2984
5.647.3.7 set	2984
5.647.4 Member Function Documentation	2985

5.647.4.1 begin	2985
5.647.4.2 cbegin	2985
5.647.4.3 cend	2985
5.647.4.4 clear	2986
5.647.4.5 count	2986
5.647.4.6 crbegin	2986
5.647.4.7 crend	2986
5.647.4.8 empty	2987
5.647.4.9 end	2987
5.647.4.10equal_range	2987
5.647.4.11lequal_range	2988
5.647.4.12erase	2988
5.647.4.13erase	2989
5.647.4.14erase	2989
5.647.4.15find	2989
5.647.4.16find	2990
5.647.4.17get_allocator	2990
5.647.4.18insert	2990
5.647.4.19insert	2991
5.647.4.20insert	2992
5.647.4.21insert	2992
5.647.4.22key_comp	2992
5.647.4.23lower_bound	2992
5.647.4.24lower_bound	2993
5.647.4.25max_size	2993
5.647.4.26operator=	2994
5.647.4.27operator=	2994
5.647.4.28operator=	2994
5.647.4.29rbegin	2995
5.647.4.30rend	2995
5.647.4.31size	2995

5.647.4.32	swap	2995
5.647.4.33	upper_bound	2996
5.647.4.34	upper_bound	2996
5.647.4.35	value_comp	2996
5.648	std::shared_future< _Res > Class Template Reference	2997
5.648.1	Detailed Description	2998
5.648.2	Constructor & Destructor Documentation	2998
5.648.2.1	shared_future	2998
5.648.2.2	shared_future	2998
5.648.2.3	shared_future	2998
5.648.3	Member Function Documentation	2999
5.648.3.1	_M_get_result	2999
5.648.3.2	get	2999
5.649	std::shared_future< _Res & > Class Template Reference	2999
5.649.1	Detailed Description	3001
5.649.2	Constructor & Destructor Documentation	3001
5.649.2.1	shared_future	3001
5.649.2.2	shared_future	3001
5.649.2.3	shared_future	3001
5.649.3	Member Function Documentation	3002
5.649.3.1	_M_get_result	3002
5.649.3.2	get	3002
5.650	std::shared_future< void > Class Template Reference	3002
5.650.1	Detailed Description	3004
5.650.2	Constructor & Destructor Documentation	3004
5.650.2.1	shared_future	3004
5.650.2.2	shared_future	3004
5.650.2.3	shared_future	3004
5.650.3	Member Function Documentation	3005
5.650.3.1	_M_get_result	3005
5.651	std::shared_ptr< _Tp > Class Template Reference	3005

5.651.1 Detailed Description	3006
5.651.2 Constructor & Destructor Documentation	3007
5.651.2.1 shared_ptr	3007
5.651.2.2 shared_ptr	3007
5.651.2.3 shared_ptr	3007
5.651.2.4 shared_ptr	3008
5.651.2.5 shared_ptr	3008
5.651.2.6 shared_ptr	3009
5.651.2.7 shared_ptr	3009
5.651.2.8 shared_ptr	3010
5.651.2.9 shared_ptr	3010
5.651.2.10 shared_ptr	3011
5.651.2.11 shared_ptr	3011
5.651.2.12 shared_ptr	3011
5.651.3 Friends And Related Function Documentation	3012
5.651.3.1 allocate_shared	3012
5.652 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference	3012
5.652.1 Detailed Description	3014
5.652.2 Member Typedef Documentation	3014
5.652.2.1 result_type	3014
5.652.3 Constructor & Destructor Documentation	3014
5.652.3.1 shuffle_order_engine	3014
5.652.3.2 shuffle_order_engine	3014
5.652.3.3 shuffle_order_engine	3015
5.652.3.4 shuffle_order_engine	3015
5.652.3.5 shuffle_order_engine	3015
5.652.4 Member Function Documentation	3016
5.652.4.1 base	3016
5.652.4.2 discard	3016
5.652.4.3 max	3016

5.652.4.4 min	3016
5.652.4.5 operator()	3017
5.652.4.6 seed	3017
5.652.4.7 seed	3017
5.652.4.8 seed	3017
5.652.5 Friends And Related Function Documentation	3018
5.652.5.1 operator<<	3018
5.652.5.2 operator==	3018
5.652.5.3 operator>>	3019
5.653std::slice Class Reference	3019
5.653.1 Detailed Description	3019
5.654std::slice_array< _Tp > Class Template Reference	3020
5.654.1 Detailed Description	3021
5.654.2 Member Function Documentation	3021
5.654.2.1 operator%=	3021
5.654.2.2 operator&=	3022
5.654.2.3 operator*=	3022
5.654.2.4 operator+=	3022
5.654.2.5 operator-=	3022
5.654.2.6 operator/=	3022
5.654.2.7 operator<<=	3022
5.654.2.8 operator>>=	3022
5.654.2.9 operator^=	3022
5.654.2.10operator =	3023
5.655std::stack< _Tp, _Sequence > Class Template Reference	3023
5.655.1 Detailed Description	3024
5.655.2 Constructor & Destructor Documentation	3025
5.655.2.1 stack	3025
5.655.3 Member Function Documentation	3025
5.655.3.1 empty	3025
5.655.3.2 pop	3025

5.655.3.3 push	3025
5.655.3.4 size	3026
5.655.3.5 top	3026
5.655.3.6 top	3026
5.656std::student_t_distribution< _RealType > Class Template Reference .	3026
5.656.1 Detailed Description	3027
5.656.2 Member Typedef Documentation	3028
5.656.2.1 result_type	3028
5.656.3 Member Function Documentation	3028
5.656.3.1 max	3028
5.656.3.2 min	3028
5.656.3.3 operator()	3028
5.656.3.4 param	3028
5.656.3.5 param	3029
5.656.3.6 reset	3029
5.656.4 Friends And Related Function Documentation	3029
5.656.4.1 operator<<	3029
5.656.4.2 operator==	3030
5.656.4.3 operator>>	3030
5.657std::student_t_distribution< _RealType >::param_type Struct Reference	3030
5.657.1 Detailed Description	3031
5.658std::sub_match< _BiIter > Class Template Reference	3032
5.658.1 Detailed Description	3033
5.658.2 Member Typedef Documentation	3033
5.658.2.1 first_type	3033
5.658.2.2 second_type	3033
5.658.3 Member Function Documentation	3034
5.658.3.1 compare	3034
5.658.3.2 compare	3034
5.658.3.3 compare	3035
5.658.3.4 length	3035

5.658.3.5 operator string_type	3035
5.658.3.6 str	3035
5.658.4 Member Data Documentation	3036
5.658.4.1 first	3036
5.658.4.2 second	3036
5.659std::system_error Class Reference	3036
5.659.1 Detailed Description	3037
5.659.2 Member Function Documentation	3037
5.659.2.1 what	3037
5.660std::thread Class Reference	3038
5.660.1 Detailed Description	3039
5.660.2 Member Function Documentation	3039
5.660.2.1 native_handle	3039
5.661std::thread::id Class Reference	3039
5.661.1 Detailed Description	3039
5.662std::time_base Class Reference	3040
5.662.1 Detailed Description	3040
5.663std::time_get< _CharT, _InIter > Class Template Reference	3041
5.663.1 Detailed Description	3043
5.663.2 Member Typedef Documentation	3043
5.663.2.1 char_type	3043
5.663.2.2 iter_type	3044
5.663.3 Constructor & Destructor Documentation	3044
5.663.3.1 time_get	3044
5.663.3.2 ~time_get	3044
5.663.4 Member Function Documentation	3044
5.663.4.1 date_order	3044
5.663.4.2 do_date_order	3045
5.663.4.3 do_get_date	3045
5.663.4.4 do_get_monthname	3046
5.663.4.5 do_get_time	3047

5.663.4.6 do_get_weekday	3047
5.663.4.7 do_get_year	3048
5.663.4.8 get_date	3049
5.663.4.9 get_monthname	3049
5.663.4.10get_time	3050
5.663.4.11get_weekday	3051
5.663.4.12get_year	3051
5.663.5 Member Data Documentation	3052
5.663.5.1 id	3052
5.664std::time_get_byname< _CharT, _InIter > Class Template Reference	3052
5.664.1 Detailed Description	3055
5.664.2 Member Typedef Documentation	3055
5.664.2.1 char_type	3055
5.664.2.2 iter_type	3055
5.664.3 Member Function Documentation	3055
5.664.3.1 date_order	3055
5.664.3.2 do_date_order	3056
5.664.3.3 do_get_date	3056
5.664.3.4 do_get_monthname	3057
5.664.3.5 do_get_time	3057
5.664.3.6 do_get_weekday	3058
5.664.3.7 do_get_year	3059
5.664.3.8 get_date	3059
5.664.3.9 get_monthname	3060
5.664.3.10get_time	3061
5.664.3.11get_weekday	3061
5.664.3.12get_year	3062
5.664.4 Member Data Documentation	3063
5.664.4.1 id	3063
5.665std::time_put< _CharT, _OutIter > Class Template Reference	3063
5.665.1 Detailed Description	3065

5.665.2 Member Typedef Documentation	3065
5.665.2.1 char_type	3065
5.665.2.2 iter_type	3066
5.665.3 Constructor & Destructor Documentation	3066
5.665.3.1 time_put	3066
5.665.3.2 ~time_put	3066
5.665.4 Member Function Documentation	3066
5.665.4.1 do_put	3066
5.665.4.2 put	3067
5.665.4.3 put	3068
5.665.5 Member Data Documentation	3068
5.665.5.1 id	3068
5.666std::time_put_byname< _CharT, _OutIter > Class Template Reference	3069
5.666.1 Detailed Description	3070
5.666.2 Member Typedef Documentation	3070
5.666.2.1 char_type	3070
5.666.2.2 iter_type	3070
5.666.3 Member Function Documentation	3071
5.666.3.1 do_put	3071
5.666.3.2 put	3071
5.666.3.3 put	3072
5.666.4 Member Data Documentation	3073
5.666.4.1 id	3073
5.667std::timed_mutex Class Reference	3073
5.667.1 Detailed Description	3073
5.668std::tr1::__is_member_pointer_helper< _Tp > Struct Template Refer-	
ence	3074
5.668.1 Detailed Description	3074
5.669std::tr1::add_const< _Tp > Struct Template Reference	3075
5.669.1 Detailed Description	3075
5.670std::tr1::add_cv< _Tp > Struct Template Reference	3075

5.670.1 Detailed Description	3075
5.671std::tr1::add_pointer< _Tp > Struct Template Reference	3076
5.671.1 Detailed Description	3076
5.672std::tr1::add_volatile< _Tp > Struct Template Reference	3076
5.672.1 Detailed Description	3076
5.673std::tr1::alignment_of< _Tp > Struct Template Reference	3077
5.673.1 Detailed Description	3077
5.674std::tr1::array< _Tp, _Nm > Struct Template Reference	3078
5.674.1 Detailed Description	3079
5.675std::tr1::bad_weak_ptr Class Reference	3079
5.675.1 Detailed Description	3080
5.675.2 Member Function Documentation	3080
5.675.2.1 what	3080
5.676std::tr1::extent< typename, _UInt > Struct Template Reference	3081
5.676.1 Detailed Description	3081
5.677std::tr1::has_virtual_destructor< _Tp > Struct Template Reference	3082
5.677.1 Detailed Description	3083
5.678std::tr1::integral_constant< _Tp, __v > Struct Template Reference	3083
5.678.1 Detailed Description	3085
5.679std::tr1::is_abstract< _Tp > Struct Template Reference	3085
5.679.1 Detailed Description	3086
5.680std::tr1::is_arithmetic< _Tp > Struct Template Reference	3086
5.680.1 Detailed Description	3087
5.681std::tr1::is_array< typename > Struct Template Reference	3087
5.681.1 Detailed Description	3088
5.682std::tr1::is_class< _Tp > Struct Template Reference	3088
5.682.1 Detailed Description	3089
5.683std::tr1::is_compound< _Tp > Struct Template Reference	3089
5.683.1 Detailed Description	3090
5.684std::tr1::is_const< typename > Struct Template Reference	3090
5.684.1 Detailed Description	3091

5.685std::tr1::is_empty< _Tp > Struct Template Reference	3091
5.685.1 Detailed Description	3092
5.686std::tr1::is_enum< _Tp > Struct Template Reference	3093
5.686.1 Detailed Description	3093
5.687std::tr1::is_floating_point< _Tp > Struct Template Reference	3094
5.687.1 Detailed Description	3094
5.688std::tr1::is_function< typename > Struct Template Reference	3095
5.688.1 Detailed Description	3095
5.689std::tr1::is_fundamental< _Tp > Struct Template Reference	3096
5.689.1 Detailed Description	3096
5.690std::tr1::is_integral< _Tp > Struct Template Reference	3096
5.690.1 Detailed Description	3097
5.691std::tr1::is_member_function_pointer< _Tp > Struct Template Reference	3097
5.691.1 Detailed Description	3098
5.692std::tr1::is_member_object_pointer< _Tp > Struct Template Reference	3098
5.692.1 Detailed Description	3099
5.693std::tr1::is_object< _Tp > Struct Template Reference	3099
5.693.1 Detailed Description	3099
5.694std::tr1::is_pointer< _Tp > Struct Template Reference	3100
5.694.1 Detailed Description	3100
5.695std::tr1::is_polymorphic< _Tp > Struct Template Reference	3100
5.695.1 Detailed Description	3101
5.696std::tr1::is_same< typename, typename > Struct Template Reference	3102
5.696.1 Detailed Description	3102
5.697std::tr1::is_scalar< _Tp > Struct Template Reference	3103
5.697.1 Detailed Description	3103
5.698std::tr1::is_union< _Tp > Struct Template Reference	3103
5.698.1 Detailed Description	3104
5.699std::tr1::is_void< _Tp > Struct Template Reference	3105
5.699.1 Detailed Description	3105

5.700std::tr1::is_volatile< typename > Struct Template Reference	3105
5.700.1 Detailed Description	3106
5.701std::tr1::rank< typename > Struct Template Reference	3106
5.701.1 Detailed Description	3107
5.702std::tr1::remove_all_extents< _Tp > Struct Template Reference	3108
5.702.1 Detailed Description	3108
5.703std::tr1::remove_const< _Tp > Struct Template Reference	3108
5.703.1 Detailed Description	3108
5.704std::tr1::remove_cv< _Tp > Struct Template Reference	3109
5.704.1 Detailed Description	3109
5.705std::tr1::remove_extent< _Tp > Struct Template Reference	3109
5.705.1 Detailed Description	3109
5.706std::tr1::remove_pointer< _Tp > Struct Template Reference	3110
5.706.1 Detailed Description	3110
5.707std::tr1::remove_volatile< _Tp > Struct Template Reference	3110
5.707.1 Detailed Description	3110
5.708std::try_to_lock_t Struct Reference	3111
5.708.1 Detailed Description	3111
5.709std::tuple< _Elements > Class Template Reference	3111
5.709.1 Detailed Description	3112
5.710std::tuple< _T1, _T2 > Class Template Reference	3112
5.710.1 Detailed Description	3113
5.711std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference	3113
5.711.1 Detailed Description	3113
5.712std::tuple_element< __i, tuple< _Head, _Tail...> > Struct Template Reference	3114
5.712.1 Detailed Description	3114
5.713std::tuple_size< tuple< _Elements...> > Struct Template Reference	3114
5.713.1 Detailed Description	3114
5.714std::type_info Class Reference	3115
5.714.1 Detailed Description	3115

5.714.2 Constructor & Destructor Documentation	3116
5.714.2.1 ~type_info	3116
5.714.3 Member Function Documentation	3116
5.714.3.1 name	3116
5.715std::unary_function< _Arg, _Result > Struct Template Reference . .	3118
5.715.1 Detailed Description	3119
5.715.2 Member Typedef Documentation	3119
5.715.2.1 argument_type	3119
5.715.2.2 result_type	3119
5.716std::unary_negate< _Predicate > Class Template Reference	3119
5.716.1 Detailed Description	3120
5.716.2 Member Typedef Documentation	3121
5.716.2.1 argument_type	3121
5.716.2.2 result_type	3121
5.717std::underflow_error Class Reference	3122
5.717.1 Detailed Description	3122
5.717.2 Member Function Documentation	3122
5.717.2.1 what	3122
5.718std::uniform_int_distribution< _IntType > Class Template Reference	3123
5.718.1 Detailed Description	3124
5.718.2 Member Typedef Documentation	3124
5.718.2.1 result_type	3124
5.718.3 Constructor & Destructor Documentation	3124
5.718.3.1 uniform_int_distribution	3124
5.718.4 Member Function Documentation	3124
5.718.4.1 max	3124
5.718.4.2 min	3125
5.718.4.3 operator()	3125
5.718.4.4 param	3125
5.718.4.5 param	3125
5.718.4.6 reset	3126

5.719	<code>std::uniform_int_distribution< _IntType >::param_type</code> Struct Reference	3126
5.719.1	Detailed Description	3126
5.720	<code>std::uniform_real_distribution< _RealType ></code> Class Template Reference	3127
5.720.1	Detailed Description	3127
5.720.2	Member Typedef Documentation	3128
5.720.2.1	<code>result_type</code>	3128
5.720.3	Constructor & Destructor Documentation	3128
5.720.3.1	<code>uniform_real_distribution</code>	3128
5.720.4	Member Function Documentation	3128
5.720.4.1	<code>max</code>	3128
5.720.4.2	<code>min</code>	3128
5.720.4.3	<code>operator()</code>	3129
5.720.4.4	<code>param</code>	3129
5.720.4.5	<code>param</code>	3129
5.720.4.6	<code>reset</code>	3129
5.721	<code>std::uniform_real_distribution< _RealType >::param_type</code> Struct Reference	3130
5.721.1	Detailed Description	3130
5.722	<code>std::unique_lock< _Mutex ></code> Class Template Reference	3130
5.722.1	Detailed Description	3131
5.723	<code>std::unique_ptr< _Tp, _Tp_Deleter ></code> Class Template Reference	3132
5.723.1	Detailed Description	3133
5.724	<code>std::unique_ptr< _Tp[], _Tp_Deleter ></code> Class Template Reference	3133
5.724.1	Detailed Description	3134
5.725	<code>std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc ></code> Class Template Reference	3134
5.725.1	Detailed Description	3137
5.726	<code>std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc ></code> Class Template Reference	3137
5.726.1	Detailed Description	3139
5.727	<code>std::unordered_multiset< _Value, _Hash, _Pred, _Alloc ></code> Class Template Reference	3140

5.727.1 Detailed Description	3142
5.728std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3143
5.728.1 Detailed Description	3145
5.729std::valarray< _Tp > Class Template Reference	3145
5.729.1 Detailed Description	3148
5.729.2 Constructor & Destructor Documentation	3149
5.729.2.1 valarray	3149
5.730std::vector< _Tp, _Alloc > Class Template Reference	3149
5.730.1 Detailed Description	3153
5.730.2 Constructor & Destructor Documentation	3153
5.730.2.1 vector	3153
5.730.2.2 vector	3154
5.730.2.3 vector	3154
5.730.2.4 vector	3154
5.730.2.5 vector	3155
5.730.2.6 vector	3155
5.730.2.7 vector	3155
5.730.2.8 vector	3156
5.730.2.9 ~vector	3156
5.730.3 Member Function Documentation	3156
5.730.3.1 _M_allocate_and_copy	3156
5.730.3.2 _M_range_check	3157
5.730.3.3 assign	3157
5.730.3.4 assign	3157
5.730.3.5 assign	3158
5.730.3.6 at	3158
5.730.3.7 at	3158
5.730.3.8 back	3159
5.730.3.9 back	3159
5.730.3.10begin	3159

5.730.3.1lbegin	3160
5.730.3.12capacity	3160
5.730.3.13cbegin	3160
5.730.3.14cend	3160
5.730.3.15clear	3161
5.730.3.16crbegin	3161
5.730.3.17crend	3161
5.730.3.18data	3161
5.730.3.19emplace	3161
5.730.3.20empty	3162
5.730.3.2lend	3162
5.730.3.22end	3162
5.730.3.23erase	3163
5.730.3.24erase	3163
5.730.3.25front	3164
5.730.3.26front	3164
5.730.3.27insert	3164
5.730.3.28insert	3164
5.730.3.29insert	3165
5.730.3.30insert	3165
5.730.3.3linsert	3166
5.730.3.32max_size	3166
5.730.3.33operator=	3166
5.730.3.34operator=	3167
5.730.3.35operator=	3167
5.730.3.36operator[]	3168
5.730.3.37operator[]	3168
5.730.3.38pop_back	3168
5.730.3.39push_back	3169
5.730.3.40rbegin	3169
5.730.3.4lrbegin	3169

5.730.3.42	rend	3169
5.730.3.43	rend	3170
5.730.3.44	reserve	3170
5.730.3.45	resize	3170
5.730.3.46	resize	3171
5.730.3.47	shrink_to_fit	3171
5.730.3.48	size	3171
5.730.3.49	swap	3172
5.731	std::vector< bool, _Alloc > Class Template Reference	3172
5.731.1	Detailed Description	3176
5.732	std::weak_ptr< _Tp > Class Template Reference	3176
5.732.1	Detailed Description	3177
5.733	std::weibull_distribution< _RealType > Class Template Reference	3177
5.733.1	Detailed Description	3178
5.733.2	Member Typedef Documentation	3178
5.733.2.1	result_type	3178
5.733.3	Member Function Documentation	3178
5.733.3.1	a	3178
5.733.3.2	b	3178
5.733.3.3	max	3178
5.733.3.4	min	3179
5.733.3.5	operator()	3179
5.733.3.6	param	3179
5.733.3.7	param	3179
5.733.3.8	reset	3180
5.734	std::weibull_distribution< _RealType >::param_type Struct Reference	3180
5.734.1	Detailed Description	3180
6	File Documentation	3181
6.1	algo.h File Reference	3181
6.1.1	Detailed Description	3195

6.2	algbase.h File Reference	3195
6.2.1	Detailed Description	3197
6.3	algorithm File Reference	3197
6.3.1	Detailed Description	3197
6.4	algorithm File Reference	3197
6.4.1	Detailed Description	3199
6.5	algorithm File Reference	3199
6.5.1	Detailed Description	3199
6.6	algorithmfwd.h File Reference	3199
6.6.1	Detailed Description	3206
6.7	algorithmfwd.h File Reference	3206
6.7.1	Detailed Description	3218
6.8	allocator.h File Reference	3219
6.8.1	Detailed Description	3219
6.9	array File Reference	3219
6.9.1	Detailed Description	3219
6.10	array File Reference	3220
6.10.1	Detailed Description	3221
6.11	array_allocator.h File Reference	3221
6.11.1	Detailed Description	3221
6.12	assoc_container.hpp File Reference	3222
6.12.1	Detailed Description	3223
6.13	atomic File Reference	3223
6.13.1	Detailed Description	3227
6.14	atomic_0.h File Reference	3227
6.14.1	Detailed Description	3228
6.15	atomic_2.h File Reference	3228
6.15.1	Detailed Description	3228
6.16	atomic_base.h File Reference	3228
6.16.1	Detailed Description	3230
6.17	atomic_word.h File Reference	3230

6.17.1 Detailed Description	3230
6.18 atomicfwd_c.h File Reference	3230
6.18.1 Detailed Description	3231
6.19 atomicfwd_cxx.h File Reference	3232
6.19.1 Detailed Description	3232
6.20 atomicity.h File Reference	3232
6.20.1 Detailed Description	3233
6.21 auto_ptr.h File Reference	3233
6.21.1 Detailed Description	3234
6.22 balanced_quicksort.h File Reference	3234
6.22.1 Detailed Description	3235
6.23 base.h File Reference	3235
6.23.1 Detailed Description	3237
6.24 base.h File Reference	3237
6.24.1 Detailed Description	3237
6.25 basic_file.h File Reference	3237
6.25.1 Detailed Description	3237
6.26 basic_ios.h File Reference	3238
6.26.1 Detailed Description	3238
6.27 basic_ios.tcc File Reference	3238
6.27.1 Detailed Description	3238
6.28 basic_iterator.h File Reference	3239
6.28.1 Detailed Description	3239
6.29 basic_string.h File Reference	3239
6.29.1 Detailed Description	3242
6.30 basic_string.tcc File Reference	3243
6.30.1 Detailed Description	3243
6.31 basic_types.hpp File Reference	3243
6.31.1 Detailed Description	3244
6.32 binders.h File Reference	3244
6.32.1 Detailed Description	3245

6.33	bitmap_allocator.h File Reference	3245
6.33.1	Detailed Description	3246
6.33.2	Define Documentation	3246
6.33.2.1	_BALLOC_ALIGN_BYTES	3246
6.34	bitset File Reference	3246
6.34.1	Detailed Description	3247
6.35	bitset File Reference	3248
6.35.1	Detailed Description	3248
6.36	bitset File Reference	3249
6.36.1	Detailed Description	3249
6.37	boost_concept_check.h File Reference	3250
6.37.1	Detailed Description	3250
6.38	boost_sp_counted_base.h File Reference	3251
6.38.1	Detailed Description	3251
6.39	c++0x_warning.h File Reference	3251
6.39.1	Detailed Description	3251
6.40	c++allocator.h File Reference	3251
6.40.1	Detailed Description	3252
6.41	c++config.h File Reference	3252
6.41.1	Detailed Description	3257
6.42	c++io.h File Reference	3257
6.42.1	Detailed Description	3257
6.43	c++locale.h File Reference	3257
6.43.1	Detailed Description	3258
6.44	c++locale_internal.h File Reference	3258
6.44.1	Detailed Description	3258
6.45	cassert File Reference	3258
6.45.1	Detailed Description	3258
6.46	ccomplex File Reference	3259
6.46.1	Detailed Description	3259
6.47	ccomplex File Reference	3259

6.47.1 Detailed Description	3259
6.48 cctype File Reference	3259
6.48.1 Detailed Description	3259
6.49 cctype File Reference	3259
6.49.1 Detailed Description	3260
6.50 cctype File Reference	3260
6.50.1 Detailed Description	3260
6.51 cerrno File Reference	3260
6.51.1 Detailed Description	3260
6.52 cenv File Reference	3260
6.52.1 Detailed Description	3260
6.53 cenv File Reference	3261
6.53.1 Detailed Description	3261
6.54 cenv File Reference	3261
6.54.1 Detailed Description	3261
6.55 cfloat File Reference	3261
6.55.1 Detailed Description	3261
6.56 cfloat File Reference	3262
6.56.1 Detailed Description	3262
6.57 char_traits.h File Reference	3262
6.57.1 Detailed Description	3263
6.58 checkers.h File Reference	3263
6.58.1 Detailed Description	3263
6.59 chrono File Reference	3263
6.59.1 Detailed Description	3266
6.60 cinttypes File Reference	3267
6.60.1 Detailed Description	3267
6.61 cinttypes File Reference	3267
6.61.1 Detailed Description	3267
6.62 cinttypes File Reference	3267
6.62.1 Detailed Description	3267

6.63	ciso646 File Reference	3267
6.63.1	Detailed Description	3267
6.64	climits File Reference	3268
6.64.1	Detailed Description	3268
6.65	climits File Reference	3268
6.65.1	Detailed Description	3268
6.66	clocale File Reference	3269
6.66.1	Detailed Description	3269
6.67	cmath File Reference	3269
6.67.1	Detailed Description	3272
6.68	cmath File Reference	3272
6.68.1	Detailed Description	3275
6.69	cmath File Reference	3276
6.69.1	Detailed Description	3276
6.70	cmath.tcc File Reference	3276
6.70.1	Detailed Description	3276
6.71	codecvt.h File Reference	3276
6.71.1	Detailed Description	3277
6.72	codecvt_specializations.h File Reference	3277
6.72.1	Detailed Description	3278
6.73	compatibility.h File Reference	3278
6.73.1	Detailed Description	3278
6.74	compatibility.h File Reference	3278
6.74.1	Detailed Description	3279
6.75	compiletime_settings.h File Reference	3279
6.75.1	Detailed Description	3279
6.75.2	Define Documentation	3280
6.75.2.1	_GLIBCXX_ASSERTIONS	3280
6.75.2.2	_GLIBCXX_CALL	3280
6.75.2.3	_GLIBCXX_RANDOM_SHUFFLE_- CONSIDER_L1	3280

6.75.2.4	_GLIBCXX_RANDOM_SHUFFLE_- CONSIDER_TLB	3281
6.75.2.5	_GLIBCXX_SCALE_DOWN_FPU	3281
6.75.2.6	_GLIBCXX_VERBOSE_LEVEL	3281
6.76	complex File Reference	3281
6.76.1	Detailed Description	3285
6.77	complex File Reference	3285
6.77.1	Detailed Description	3286
6.78	complex File Reference	3286
6.78.1	Detailed Description	3287
6.79	complex.h File Reference	3287
6.79.1	Detailed Description	3287
6.80	concept_check.h File Reference	3288
6.80.1	Detailed Description	3288
6.81	concurrency.h File Reference	3288
6.81.1	Detailed Description	3289
6.82	cond_dealtor.hpp File Reference	3289
6.82.1	Detailed Description	3289
6.83	condition_variable File Reference	3289
6.83.1	Detailed Description	3290
6.84	constructors_destructor_fn_imps.hpp File Reference	3290
6.84.1	Detailed Description	3291
6.85	container_base_dispatch.hpp File Reference	3291
6.85.1	Detailed Description	3291
6.86	cpp_type_traits.h File Reference	3291
6.86.1	Detailed Description	3291
6.87	cpu_defines.h File Reference	3292
6.87.1	Detailed Description	3292
6.88	csetjmp File Reference	3292
6.88.1	Detailed Description	3292
6.89	csignal File Reference	3292

6.89.1 Detailed Description	3292
6.90 cstdarg File Reference	3293
6.90.1 Detailed Description	3293
6.91 cstdarg File Reference	3293
6.91.1 Detailed Description	3293
6.92 cstdbool File Reference	3293
6.92.1 Detailed Description	3293
6.93 cstdbool File Reference	3294
6.93.1 Detailed Description	3294
6.94 cstddef File Reference	3294
6.94.1 Detailed Description	3294
6.95 cstdint File Reference	3294
6.95.1 Detailed Description	3294
6.96 cstdint File Reference	3294
6.96.1 Detailed Description	3294
6.97 cstdint File Reference	3295
6.97.1 Detailed Description	3295
6.98 cstdio File Reference	3295
6.98.1 Detailed Description	3295
6.99 cstdio File Reference	3295
6.99.1 Detailed Description	3296
6.100 cstdio File Reference	3296
6.100.1 Detailed Description	3296
6.101 cstdlib File Reference	3296
6.101.1 Detailed Description	3297
6.102 cstdlib File Reference	3297
6.102.1 Detailed Description	3297
6.103 cstdlib File Reference	3297
6.103.1 Detailed Description	3297
6.104 cstring File Reference	3297
6.104.1 Detailed Description	3298

6.105ctgmath File Reference	3298
6.105.1 Detailed Description	3298
6.106ctgmath File Reference	3298
6.106.1 Detailed Description	3298
6.107ctime File Reference	3298
6.107.1 Detailed Description	3299
6.108ctime File Reference	3299
6.108.1 Detailed Description	3299
6.109ctype_base.h File Reference	3299
6.109.1 Detailed Description	3299
6.110ctype_inline.h File Reference	3300
6.110.1 Detailed Description	3300
6.111ctype_noninline.h File Reference	3300
6.111.1 Detailed Description	3300
6.112wchar File Reference	3300
6.112.1 Detailed Description	3300
6.113wchar File Reference	3301
6.113.1 Detailed Description	3301
6.114wchar File Reference	3301
6.114.1 Detailed Description	3301
6.115cwctype File Reference	3301
6.115.1 Detailed Description	3302
6.116cwctype File Reference	3302
6.116.1 Detailed Description	3302
6.117cwctype File Reference	3302
6.117.1 Detailed Description	3302
6.118cxxabi-forced.h File Reference	3303
6.118.1 Detailed Description	3303
6.119cxxabi.h File Reference	3303
6.119.1 Detailed Description	3305
6.120cxxabi_tweaks.h File Reference	3305

6.120.1 Detailed Description	3305
6.121 debug.h File Reference	3305
6.121.1 Detailed Description	3306
6.122 debug_allocator.h File Reference	3306
6.122.1 Detailed Description	3307
6.123 debug_map_base.hpp File Reference	3307
6.123.1 Detailed Description	3307
6.124 decimal File Reference	3307
6.124.1 Detailed Description	3319
6.125 deque File Reference	3319
6.125.1 Detailed Description	3319
6.126 deque File Reference	3319
6.126.1 Detailed Description	3320
6.127 deque File Reference	3320
6.127.1 Detailed Description	3321
6.128 deque.tcc File Reference	3321
6.128.1 Detailed Description	3322
6.129 enc_filebuf.h File Reference	3322
6.129.1 Detailed Description	3322
6.130 equally_split.h File Reference	3323
6.130.1 Detailed Description	3323
6.131 error_constants.h File Reference	3323
6.131.1 Detailed Description	3324
6.132 exception File Reference	3324
6.132.1 Detailed Description	3325
6.133 exception.hpp File Reference	3325
6.133.1 Detailed Description	3326
6.134 exception_ptr.h File Reference	3326
6.134.1 Detailed Description	3326
6.135 extptr_allocator.h File Reference	3327
6.135.1 Detailed Description	3327

6.136features.h File Reference	3327
6.136.1 Detailed Description	3328
6.136.2 Define Documentation	3328
6.136.2.1 _GLIBCXX_BAL_QUICKSORT	3328
6.136.2.2 _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS	3328
6.136.2.3 _GLIBCXX_FIND_EQUAL_SPLIT	3328
6.136.2.4 _GLIBCXX_FIND_GROWING_BLOCKS	3329
6.136.2.5 _GLIBCXX_MERGESORT	3329
6.136.2.6 _GLIBCXX_QUICKSORT	3329
6.136.2.7 _GLIBCXX_TREE_DYNAMIC_BALANCING	3329
6.136.2.8 _GLIBCXX_TREE_FULL_COPY	3330
6.136.2.9 _GLIBCXX_TREE_INITIAL_SPLITTING	3330
6.137fenv.h File Reference	3330
6.137.1 Detailed Description	3330
6.138find.h File Reference	3330
6.138.1 Detailed Description	3331
6.139find_selectors.h File Reference	3331
6.139.1 Detailed Description	3332
6.140for_each.h File Reference	3332
6.140.1 Detailed Description	3332
6.141for_each_selectors.h File Reference	3333
6.141.1 Detailed Description	3334
6.142formatter.h File Reference	3335
6.142.1 Detailed Description	3335
6.143forward_list.h File Reference	3336
6.143.1 Detailed Description	3337
6.144forward_list.tcc File Reference	3337
6.144.1 Detailed Description	3337
6.145fstream File Reference	3338
6.145.1 Detailed Description	3338
6.146fstream.tcc File Reference	3339

6.146.1 Detailed Description	3339
6.147funcexcept.h File Reference	3339
6.147.1 Detailed Description	3340
6.148functional File Reference	3340
6.148.1 Detailed Description	3343
6.149functional File Reference	3344
6.149.1 Detailed Description	3345
6.150functional_hash.h File Reference	3345
6.150.1 Detailed Description	3346
6.151functions.h File Reference	3346
6.151.1 Detailed Description	3348
6.152future File Reference	3348
6.152.1 Detailed Description	3351
6.153gslice.h File Reference	3351
6.153.1 Detailed Description	3351
6.154gslice_array.h File Reference	3351
6.154.1 Detailed Description	3352
6.155hash_fun.h File Reference	3352
6.155.1 Detailed Description	3352
6.156hash_map File Reference	3352
6.156.1 Detailed Description	3353
6.157hash_policy.hpp File Reference	3353
6.157.1 Detailed Description	3354
6.158hash_set File Reference	3354
6.158.1 Detailed Description	3355
6.159hashtable.h File Reference	3355
6.159.1 Detailed Description	3356
6.160hashtable.h File Reference	3356
6.160.1 Detailed Description	3356
6.161hashtable_policy.h File Reference	3357
6.161.1 Detailed Description	3357

6.162indirect_array.h File Reference	3358
6.162.1 Detailed Description	3358
6.163initializer_list File Reference	3358
6.163.1 Detailed Description	3358
6.164iomanip File Reference	3359
6.164.1 Detailed Description	3360
6.165ios File Reference	3360
6.165.1 Detailed Description	3360
6.166ios_base.h File Reference	3360
6.166.1 Detailed Description	3363
6.167iosfwd File Reference	3363
6.167.1 Detailed Description	3364
6.168iostream File Reference	3364
6.168.1 Detailed Description	3364
6.169istream File Reference	3365
6.169.1 Detailed Description	3366
6.170istream.tcc File Reference	3366
6.170.1 Detailed Description	3366
6.171iterator File Reference	3367
6.171.1 Detailed Description	3367
6.172iterator File Reference	3367
6.172.1 Detailed Description	3367
6.173iterator.h File Reference	3367
6.173.1 Detailed Description	3368
6.174iterator_tracker.h File Reference	3368
6.174.1 Detailed Description	3370
6.175limits File Reference	3370
6.175.1 Detailed Description	3372
6.176list File Reference	3372
6.176.1 Detailed Description	3372
6.177list File Reference	3373

6.177.1 Detailed Description	3373
6.178list File Reference	3374
6.178.1 Detailed Description	3374
6.179list.tcc File Reference	3375
6.179.1 Detailed Description	3375
6.180list_partition.h File Reference	3375
6.180.1 Detailed Description	3375
6.181list_update_policy.hpp File Reference	3376
6.181.1 Detailed Description	3376
6.182locale File Reference	3376
6.182.1 Detailed Description	3376
6.183locale_classes.h File Reference	3376
6.183.1 Detailed Description	3377
6.184locale_classes.tcc File Reference	3377
6.184.1 Detailed Description	3378
6.185locale_facets.h File Reference	3378
6.185.1 Detailed Description	3380
6.186locale_facets.tcc File Reference	3381
6.186.1 Detailed Description	3381
6.187locale_facets_nonio.h File Reference	3381
6.187.1 Detailed Description	3383
6.188locale_facets_nonio.tcc File Reference	3383
6.188.1 Detailed Description	3383
6.189localefwd.h File Reference	3383
6.189.1 Detailed Description	3384
6.190losertree.h File Reference	3384
6.190.1 Detailed Description	3386
6.191macros.h File Reference	3386
6.191.1 Detailed Description	3386
6.191.2 Define Documentation	3387
6.191.2.1 __glibcxx_check_erase	3387

6.191.2.2	__glibcxx_check_erase_range	3387
6.191.2.3	__glibcxx_check_heap_pred	3387
6.191.2.4	__glibcxx_check_insert	3387
6.191.2.5	__glibcxx_check_insert_range	3387
6.191.2.6	__glibcxx_check_partitioned_lower	3388
6.191.2.7	__glibcxx_check_partitioned_lower_pred	3388
6.191.2.8	__glibcxx_check_partitioned_upper_pred	3388
6.191.2.9	__glibcxx_check_sorted_pred	3388
6.191.2.10	GLIBCXX_DEBUG_VERIFY	3388
6.192	malloc_allocator.h File Reference	3389
6.192.1	Detailed Description	3389
6.193	map File Reference	3389
6.193.1	Detailed Description	3389
6.194	map File Reference	3390
6.194.1	Detailed Description	3390
6.195	map File Reference	3390
6.195.1	Detailed Description	3390
6.196	map.h File Reference	3390
6.196.1	Detailed Description	3391
6.197	map.h File Reference	3391
6.197.1	Detailed Description	3392
6.198	mask_array.h File Reference	3392
6.198.1	Detailed Description	3393
6.199	memory File Reference	3393
6.199.1	Detailed Description	3393
6.200	memory File Reference	3393
6.200.1	Detailed Description	3394
6.201	merge.h File Reference	3394
6.201.1	Detailed Description	3395
6.202	messages_members.h File Reference	3395
6.202.1	Detailed Description	3395

6.203move.h File Reference	3395
6.203.1 Detailed Description	3396
6.204mt_allocator.h File Reference	3397
6.204.1 Detailed Description	3398
6.205multimap.h File Reference	3398
6.205.1 Detailed Description	3399
6.206multimap.h File Reference	3399
6.206.1 Detailed Description	3400
6.207multiseq_selection.h File Reference	3400
6.207.1 Detailed Description	3401
6.208multiset.h File Reference	3401
6.208.1 Detailed Description	3402
6.209multiset.h File Reference	3402
6.209.1 Detailed Description	3403
6.210multiway_merge.h File Reference	3403
6.210.1 Detailed Description	3408
6.210.2 Define Documentation	3409
6.210.2.1 _GLIBCXX_PARALLEL_LENGTH	3409
6.211multiway_mergesort.h File Reference	3409
6.211.1 Detailed Description	3410
6.212mutex File Reference	3410
6.212.1 Detailed Description	3412
6.213nested_exception.h File Reference	3412
6.213.1 Detailed Description	3413
6.214new File Reference	3413
6.214.1 Detailed Description	3414
6.214.2 Function Documentation	3414
6.214.2.1 operator delete	3414
6.214.2.2 operator delete	3414
6.214.2.3 operator delete	3415
6.214.2.4 operator delete[]	3415

6.214.2.5 operator delete[]	3415
6.214.2.6 operator delete[]	3416
6.214.2.7 operator new	3416
6.214.2.8 operator new	3416
6.214.2.9 operator new	3417
6.214.2.10operator new[]	3417
6.214.2.11operator new[]	3417
6.214.2.12operator new[]	3418
6.215new_allocator.h File Reference	3418
6.215.1 Detailed Description	3419
6.216numeric File Reference	3419
6.216.1 Detailed Description	3419
6.217numeric File Reference	3419
6.217.1 Detailed Description	3419
6.218numeric File Reference	3420
6.218.1 Detailed Description	3423
6.219numeric_traits.h File Reference	3423
6.219.1 Detailed Description	3423
6.220numericfwd.h File Reference	3423
6.220.1 Detailed Description	3426
6.221omp_loop.h File Reference	3426
6.221.1 Detailed Description	3426
6.222omp_loop_static.h File Reference	3427
6.222.1 Detailed Description	3427
6.223os_defines.h File Reference	3427
6.223.1 Detailed Description	3427
6.224ostream File Reference	3428
6.224.1 Detailed Description	3429
6.225ostream.tcc File Reference	3429
6.225.1 Detailed Description	3430
6.226ostream_insert.h File Reference	3430

6.226.1 Detailed Description	3430
6.227par_loop.h File Reference	3430
6.227.1 Detailed Description	3431
6.228parallel.h File Reference	3431
6.228.1 Detailed Description	3431
6.229partial_sum.h File Reference	3431
6.229.1 Detailed Description	3432
6.230partition.h File Reference	3432
6.230.1 Detailed Description	3433
6.230.2 Define Documentation	3433
6.230.2.1 _GLIBCXX_VOLATILE	3433
6.231pod_char_traits.h File Reference	3433
6.231.1 Detailed Description	3434
6.232pointer.h File Reference	3434
6.232.1 Detailed Description	3436
6.233pool_allocator.h File Reference	3437
6.233.1 Detailed Description	3437
6.234postypes.h File Reference	3437
6.234.1 Detailed Description	3438
6.235priority_queue.hpp File Reference	3438
6.235.1 Detailed Description	3439
6.236priority_queue_base_dispatch.hpp File Reference	3439
6.236.1 Detailed Description	3439
6.237profiler.h File Reference	3439
6.237.1 Detailed Description	3442
6.238profiler_algos.h File Reference	3442
6.238.1 Detailed Description	3443
6.239profiler_hashtable_size.h File Reference	3443
6.239.1 Detailed Description	3444
6.240profiler_list_to_slist.h File Reference	3444
6.240.1 Detailed Description	3444

6.241	profiler_list_to_vector.h File Reference	3444
6.241.1	Detailed Description	3445
6.242	profiler_map_to_unordered_map.h File Reference	3445
6.242.1	Detailed Description	3446
6.243	profiler_node.h File Reference	3447
6.243.1	Detailed Description	3447
6.244	profiler_state.h File Reference	3448
6.244.1	Detailed Description	3448
6.245	profiler_trace.h File Reference	3448
6.245.1	Detailed Description	3451
6.246	profiler_vector_size.h File Reference	3451
6.246.1	Detailed Description	3451
6.247	profiler_vector_to_list.h File Reference	3451
6.247.1	Detailed Description	3452
6.248	queue File Reference	3452
6.248.1	Detailed Description	3452
6.249	queue.h File Reference	3453
6.249.1	Detailed Description	3453
6.249.2	Define Documentation	3453
6.249.2.1	_GLIBCXX_VOLATILE	3453
6.250	quicksort.h File Reference	3454
6.250.1	Detailed Description	3454
6.251	random File Reference	3454
6.251.1	Detailed Description	3454
6.252	random.h File Reference	3455
6.252.1	Detailed Description	3462
6.253	random.tcc File Reference	3462
6.253.1	Detailed Description	3467
6.254	random_number.h File Reference	3467
6.254.1	Detailed Description	3467
6.255	random_shuffle.h File Reference	3467

6.255.1 Detailed Description	3469
6.256ratio File Reference	3469
6.256.1 Detailed Description	3470
6.257rb_tree File Reference	3470
6.257.1 Detailed Description	3471
6.258rc_string_base.h File Reference	3471
6.258.1 Detailed Description	3471
6.259regex File Reference	3471
6.259.1 Detailed Description	3471
6.260regex.h File Reference	3471
6.260.1 Detailed Description	3477
6.261regex_compiler.h File Reference	3478
6.261.1 Detailed Description	3478
6.262regex_constants.h File Reference	3478
6.262.1 Detailed Description	3479
6.263regex_cursor.h File Reference	3480
6.263.1 Detailed Description	3480
6.264regex_error.h File Reference	3480
6.264.1 Detailed Description	3481
6.265regex_grep_matcher.h File Reference	3481
6.265.1 Detailed Description	3482
6.266regex_grep_matcher.tcc File Reference	3482
6.266.1 Detailed Description	3482
6.267regex_nfa.h File Reference	3482
6.267.1 Detailed Description	3483
6.268regex_nfa.tcc File Reference	3483
6.268.1 Detailed Description	3483
6.269rope File Reference	3483
6.269.1 Detailed Description	3487
6.270ropeimpl.h File Reference	3488
6.270.1 Detailed Description	3488

6.271safe_base.h File Reference	3488
6.271.1 Detailed Description	3489
6.272safe_iterator.h File Reference	3489
6.272.1 Detailed Description	3490
6.273safe_iterator.tcc File Reference	3491
6.273.1 Detailed Description	3491
6.274safe_sequence.h File Reference	3491
6.274.1 Detailed Description	3491
6.275search.h File Reference	3491
6.275.1 Detailed Description	3492
6.276set File Reference	3492
6.276.1 Detailed Description	3492
6.277set File Reference	3492
6.277.1 Detailed Description	3492
6.278set File Reference	3492
6.278.1 Detailed Description	3492
6.279set.h File Reference	3493
6.279.1 Detailed Description	3493
6.280set.h File Reference	3494
6.280.1 Detailed Description	3494
6.281set_operations.h File Reference	3495
6.281.1 Detailed Description	3495
6.282settings.h File Reference	3496
6.282.1 Detailed Description	3496
6.282.2 parallelization_decision	3496
6.282.3 Define Documentation	3497
6.282.3.1 _GLIBCXX_PARALLEL_CONDITION	3497
6.283shared_ptr.h File Reference	3497
6.283.1 Detailed Description	3499
6.284shared_ptr_base.h File Reference	3499
6.284.1 Detailed Description	3499

6.285slice_array.h File Reference	3499
6.285.1 Detailed Description	3500
6.286slist File Reference	3500
6.286.1 Detailed Description	3501
6.287sort.h File Reference	3501
6.287.1 Detailed Description	3502
6.288sso_string_base.h File Reference	3502
6.288.1 Detailed Description	3502
6.289sstream File Reference	3503
6.289.1 Detailed Description	3503
6.290sstream.tcc File Reference	3504
6.290.1 Detailed Description	3504
6.291stack File Reference	3504
6.291.1 Detailed Description	3504
6.292standard_policies.hpp File Reference	3504
6.292.1 Detailed Description	3504
6.293stdatomic.h File Reference	3505
6.293.1 Detailed Description	3505
6.294stdexcept File Reference	3505
6.294.1 Detailed Description	3505
6.295stdio_filebuf.h File Reference	3505
6.295.1 Detailed Description	3506
6.296stdio_sync_filebuf.h File Reference	3506
6.296.1 Detailed Description	3506
6.297stl_algo.h File Reference	3507
6.297.1 Detailed Description	3520
6.298stl_algobase.h File Reference	3520
6.298.1 Detailed Description	3522
6.299stl_bvector.h File Reference	3523
6.299.1 Detailed Description	3523
6.300stl_construct.h File Reference	3524

6.300.1 Detailed Description	3524
6.301stl_deque.h File Reference	3524
6.301.1 Detailed Description	3527
6.301.2 Define Documentation	3528
6.301.2.1 _GLIBCXX_DEQUE_BUF_SIZE	3528
6.302stl_function.h File Reference	3528
6.302.1 Detailed Description	3531
6.303stl_heap.h File Reference	3531
6.303.1 Detailed Description	3533
6.304stl_iterator.h File Reference	3533
6.304.1 Detailed Description	3537
6.305stl_iterator_base_funcs.h File Reference	3537
6.305.1 Detailed Description	3538
6.306stl_iterator_base_types.h File Reference	3538
6.306.1 Detailed Description	3539
6.307stl_list.h File Reference	3539
6.307.1 Detailed Description	3541
6.308stl_map.h File Reference	3541
6.308.1 Detailed Description	3542
6.309stl_multimap.h File Reference	3542
6.309.1 Detailed Description	3543
6.310stl_multiset.h File Reference	3543
6.310.1 Detailed Description	3544
6.311stl_numeric.h File Reference	3544
6.311.1 Detailed Description	3545
6.312stl_pair.h File Reference	3545
6.312.1 Detailed Description	3546
6.313stl_queue.h File Reference	3546
6.313.1 Detailed Description	3547
6.314stl_raw_storage_iter.h File Reference	3547
6.314.1 Detailed Description	3547

6.315	stl_relops.h File Reference	3547
6.315.1	Detailed Description	3548
6.316	stl_set.h File Reference	3548
6.316.1	Detailed Description	3549
6.317	stl_stack.h File Reference	3549
6.317.1	Detailed Description	3550
6.318	stl_tempbuf.h File Reference	3550
6.318.1	Detailed Description	3551
6.319	stl_tree.h File Reference	3551
6.319.1	Detailed Description	3552
6.320	stl_uninitialized.h File Reference	3553
6.320.1	Detailed Description	3555
6.321	stl_vector.h File Reference	3555
6.321.1	Detailed Description	3556
6.322	stream_iterator.h File Reference	3556
6.322.1	Detailed Description	3556
6.323	streambuf File Reference	3557
6.323.1	Detailed Description	3557
6.324	streambuf.tcc File Reference	3557
6.324.1	Detailed Description	3558
6.325	streambuf_iterator.h File Reference	3558
6.325.1	Detailed Description	3559
6.326	string File Reference	3559
6.326.1	Detailed Description	3559
6.327	string File Reference	3559
6.327.1	Detailed Description	3562
6.328	stringfwd.h File Reference	3562
6.328.1	Detailed Description	3563
6.329	system_error File Reference	3563
6.329.1	Detailed Description	3564
6.330	tag_and_trait.hpp File Reference	3564

6.330.1 Detailed Description	3566
6.331tags.h File Reference	3566
6.331.1 Detailed Description	3568
6.332tgmath.h File Reference	3568
6.332.1 Detailed Description	3568
6.333thread File Reference	3568
6.333.1 Detailed Description	3569
6.334throw_allocator.h File Reference	3570
6.334.1 Detailed Description	3572
6.335time_members.h File Reference	3572
6.335.1 Detailed Description	3572
6.336tree_policy.hpp File Reference	3573
6.336.1 Detailed Description	3573
6.337tree_trace_base.hpp File Reference	3573
6.337.1 Detailed Description	3573
6.338trie_policy.hpp File Reference	3573
6.338.1 Detailed Description	3574
6.339tuple File Reference	3574
6.339.1 Detailed Description	3576
6.340type_traits File Reference	3576
6.340.1 Detailed Description	3579
6.341type_traits File Reference	3579
6.341.1 Detailed Description	3582
6.342type_traits.h File Reference	3582
6.342.1 Detailed Description	3583
6.343type_utils.hpp File Reference	3583
6.343.1 Detailed Description	3583
6.344typeinfo File Reference	3583
6.344.1 Detailed Description	3584
6.345typelist.h File Reference	3584
6.345.1 Detailed Description	3585

6.346types.h File Reference	3585
6.346.1 Detailed Description	3586
6.347types_traits.hpp File Reference	3587
6.347.1 Detailed Description	3587
6.348unique_copy.h File Reference	3587
6.348.1 Detailed Description	3587
6.349unique_ptr.h File Reference	3588
6.349.1 Detailed Description	3589
6.350unordered_map File Reference	3589
6.350.1 Detailed Description	3589
6.351unordered_map File Reference	3589
6.351.1 Detailed Description	3590
6.352unordered_map File Reference	3590
6.352.1 Detailed Description	3592
6.353unordered_map.h File Reference	3592
6.353.1 Detailed Description	3594
6.354unordered_set File Reference	3594
6.354.1 Detailed Description	3594
6.355unordered_set File Reference	3594
6.355.1 Detailed Description	3595
6.356unordered_set File Reference	3595
6.356.1 Detailed Description	3596
6.357unordered_set.h File Reference	3596
6.357.1 Detailed Description	3598
6.358utility File Reference	3598
6.358.1 Detailed Description	3598
6.359utility File Reference	3598
6.359.1 Detailed Description	3599
6.360valarray File Reference	3599
6.360.1 Detailed Description	3604
6.361valarray_after.h File Reference	3604

6.361.1 Detailed Description	3619
6.362 valarray_array.h File Reference	3619
6.362.1 Detailed Description	3629
6.363 valarray_array.tcc File Reference	3630
6.363.1 Detailed Description	3631
6.364 valarray_before.h File Reference	3631
6.364.1 Detailed Description	3631
6.365 vector File Reference	3631
6.365.1 Detailed Description	3631
6.366 vector File Reference	3631
6.366.1 Detailed Description	3632
6.367 vector File Reference	3632
6.367.1 Detailed Description	3633
6.368 vector.tcc File Reference	3634
6.368.1 Detailed Description	3634
6.369 vstring.h File Reference	3634
6.369.1 Detailed Description	3638
6.370 vstring.tcc File Reference	3638
6.370.1 Detailed Description	3639
6.371 vstring_fwd.h File Reference	3639
6.371.1 Detailed Description	3640
6.372 vstring_util.h File Reference	3640
6.372.1 Detailed Description	3640
6.373 workstealing.h File Reference	3640
6.373.1 Detailed Description	3641

Chapter 1

Todo List

Member [__glibcxx_check_insert_range](#)([_Position](#), [_First](#), [_Last](#)) We would like to be able to check for noninterference of [_Position](#) and the range [[_First](#), [_Last](#)), but that can't (in general) be done.

Member [__gnu_cxx::distance](#)([_InputIterator](#) [__first](#), [_InputIterator](#) [__last](#), [Distance](#) & [__n](#))
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_map](#)< [_Key](#), [_Tp](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_multimap](#)< [_Key](#), [_Tp](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_multiset](#)< [_Value](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class **`__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc>`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::iota(_ForwardIter __first, _ForwardIter __last, _Tp __value)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::is_heap(_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::is_heap(_RandomAccessIterator __first, _RandomAccessIterator __last)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::is_sorted(_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::is_sorted(_ForwardIterator __first, _ForwardIterator __last)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::power(_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`__gnu_cxx::power(_Tp __x, _Integer __n)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample](#)([_InputIterator __first](#), [_InputIterator __last](#), [_RandomAccessIterator __out_first](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample](#)([_InputIterator __first](#), [_InputIterator __last](#), [_RandomAccessIterator __out_first](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample_n](#)([_ForwardIterator __first](#), [_ForwardIterator __last](#), [_OutputIterator __out](#), [const](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample_n](#)([_ForwardIterator __first](#), [_ForwardIterator __last](#), [_OutputIterator __out](#), [const](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::rb_tree](#)<[_Key](#), [_Value](#), [_KeyOfValue](#), [_Compare](#), [_Alloc](#)>

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::rope](#)<[_CharT](#), [_Alloc](#)> Doc me! See doc/doxygen/TODO and

<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::slist](#)<[_Tp](#), [_Alloc](#)> Doc me! See doc/doxygen/TODO and

<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_debug::Safe_iterator](#)<[_Iterator](#), [_Sequence](#)>::[operator->\(\)](#) const

Make this correct w.r.t. iterators that return proxies

Use [addressof\(\)](#) instead of & operator

Group **Constants** These should be constexpr.

Class **std::basic_string<_CharT, _Traits, _Alloc>**

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard(unsigned long long __z)**

Look for a faster way to do discard.

Member **std::discard_block_engine<_RandomNumberEngine, __p, __r>::max() const**

This should be constexpr.

Member **std::discard_block_engine<_RandomNumberEngine, __p, __r>::min() const**

This should be constexpr.

Member **std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::discard(unsigned long long __z)**

Look for a faster way to do discard.

Member **std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max() const**

This should be constexpr.

Member **std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min() const**

This should be constexpr.

Member **std::linear_congruential_engine<_UIntType, __a, __c, __m>::discard(unsigned long long __z)**

Look for a faster way to do discard.

Member **std::linear_congruential_engine<_UIntType, __a, __c, __m>::max() const**

This should be constexpr.

Member **std::linear_congruential_engine<_UIntType, __a, __c, __m>::min() const**

This should be constexpr.

Member **`std::match_results<_Bi_iter, _Allocator>::format(_Out_iter __out, const string_type &__fmt, regex_constants::match_flag_type __flags)`**
Implement this function.

Member **`std::match_results<_Bi_iter, _Allocator>::format(const string_type &__fmt, regex_constants::match_flag_type __flags)`**
Implement this function.

Member **`std::operator==(const match_results<_Bi_iter, _Allocator> &__m1, const match_results<_Bi_iter, _Allocator> &__m2)`**
Implement this function.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator!=(const regex_iterator &__rhs)`**
Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator*()`**
Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++()`**
Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++(int)`**
Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator->()`**
Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator=(const regex_iterator &__rhs)`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator==(const regex_iterator &__rhs)`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator()`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator(_Bi_iter __a, _Bi_iter __b,`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator(const regex_iterator &__r,`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_match(_Bi_iter __s, _Bi_iter __e, match_results<_Bi_iter, _Allocator> &__m, const`**

Implement this function.

Member **`std::regex_replace(const basic_string<_Ch_type> &__s, const basic_regex<_Ch_type, _Rx_traits>`**

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_replace`(`_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Implement this function.

Implement this function.

Member `std::regex_search`(`_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Allocator > &__m, const basic_r`

Implement this function.

Member `std::regex_search`(`const _Ch_type *__s, match_results< const _Ch_type *, _Allocator > &__m, const basic_re`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_search`(`_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_c`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_search`(`const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::m`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_search`(`const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _C`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=(const regex_token_iterator &__rhs)`

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator*()`

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`

Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator++(int)`
Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator->()`
Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator=(const regex_token_itera`
Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator==(const regex_token_iter`
Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator(_Bi_iter __a,`
Implement this function.
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator(_Bi_iter __a,`
Implement this function.
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator()`
Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator(_Bi_iter __a,`
Implement this function.
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator(const regex_t`
Implement this function.

Member **`std::regex_traits<_Ch_type>::lookup_classname`**(_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const
Implement this function.

Member **`std::regex_traits<_Ch_type>::lookup_collatename`**(_Fwd_iter __first, _Fwd_iter __last) const
Implement this function.

Member **`std::regex_traits<_Ch_type>::transform_primary`**(_Fwd_iter __first, _Fwd_iter __last) const
Implement this function.

Member **`std::reverse_iterator<_Iterator>::operator*()`** const
Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator<_Iterator>::operator+(difference_type __n)`** const
Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator<_Iterator>::operator++()`**
Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator<_Iterator>::operator++(int)`**
Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator<_Iterator>::operator+=(difference_type __n)`**
Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::reverse_iterator<_Iterator>::operator-(difference_type __n)`** const
Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator--()`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator--(int)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator--(difference_type __n)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator->() const`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator[](difference_type __n) const`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::discard(unsigned long long __z)`

Look for a faster way to do discard.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::max() const`

This should be constexpr.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::min() const`

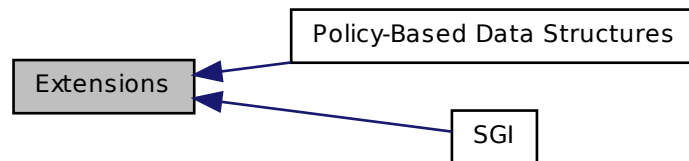
This should be constexpr.

Chapter 2

Module Documentation

2.1 Extensions

Collaboration diagram for Extensions:



Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`
Template class `__versa_string`.
Data structure managing sequences of characters and character-like objects.

Modules

- [SGI](#)

- [Policy-Based Data Structures](#)

2.1.1 Detailed Description

Components generally useful that are not part of any standard.

2.2 SGI

Collaboration diagram for SGI:



Classes

- `class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
An SGI extension .
- `struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
An SGI extension .
- `struct __gnu_cxx::constant_unary_fun< _Result, _Argument >`
An SGI extension .
- `struct __gnu_cxx::constant_void_fun< _Result >`
An SGI extension .
- `class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- `class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- `class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- `class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- `struct __gnu_cxx::project1st< _Arg1, _Arg2 >`
An SGI extension .

- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
An SGI extension .
- struct `__gnu_cxx::select2nd< _Pair >`
An SGI extension .
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`
An SGI extension .

Functions

- template<typename _Tp >
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare >
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- size_t `std::bitset::_Find_first` () const
- size_t `std::bitset::_Find_next` (size_t __prev) const
- template<class _Operation1, class _Operation2 >
unary_compose< _Operation1, _Operation2 > `__gnu_cxx::compose1` (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3 >
binary_compose< _Operation1, _Operation2, _Operation3 > `__gnu_cxx::compose2` (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result >
constant_void_fun< _Result > `__gnu_cxx::constant0` (const _Result &__val)
- template<class _Result >
constant_unary_fun< _Result, _Result > `__gnu_cxx::constant1` (const _Result &__val)
- template<class _Result >
constant_binary_fun< _Result, _Result, _Result > `__gnu_cxx::constant2` (const _Result &__val)

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __`
`first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Distance >`
`void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _`
`Distance &__n)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<typename _ForwardIter, typename _Tp >`
`void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _`
`RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`
`bool __gnu_cxx::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _`
`StrictWeakOrdering __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid-`
`op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _`
`RandomNumberGenerator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __`
`first, _InputIterator __last, _RandomAccessIterator __out_first, _`
`RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __`
`first, _InputIterator __last, _RandomAccessIterator __out_first, _`
`RandomAccessIterator __out_last)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _`
`ForwardIterator __last, _OutputIterator __out, const _Distance __n)`

- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _-`
`ForwardIterator __last, _OutputIterator __out, const _Distance __n, _-`
`RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter`
`__first, _Size __count, _ForwardIter __result)`
- `bitset< _Nb > & std::bitset::_Unchecked_set (size_t __pos)`
- `bitset< _Nb > & std::bitset::_Unchecked_set (size_t __pos, int __val)`
- `bitset< _Nb > & std::bitset::_Unchecked_reset (size_t __pos)`
- `bitset< _Nb > & std::bitset::_Unchecked_flip (size_t __pos)`
- `bool std::bitset::_Unchecked_test (size_t __pos) const`

2.2.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functions and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int templ = g1(x);
int temp2 = g2(x);
int answer = f(templ,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

2.2.2 Function Documentation

2.2.2.1 `template<typename _Tp> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c)`

Find the median of three values.

Parameters

- a* A value.
- b* A value.
- c* A value.

Returns

One of *a*, *b* or *c*.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that $1 \leq m \leq n$ then the value returned will be m . This is an SGI extension.

Definition at line 537 of file ext/algorithm.

2.2.2.2 `template<typename _Tp , typename _Compare > const _Tp&
__gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp
& __c, _Compare __comp)`

Find the median of three values using a predicate for comparison.

Parameters

a A value.
b A value.
c A value.
comp A binary predicate.

Returns

One of *a*, *b* or *c*.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that `comp(1, m)` and `comp(m, n)` are both true then the value returned will be m . This is an SGI extension.

Definition at line 571 of file ext/algorithm.

2.2.2.3 `template<size_t _Nb> size_t std::bitset<_Nb>::__Find_first () const
[inline, inherited]`

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See also

`_Find_next`

Definition at line 1303 of file bitset.

2.2.2.4 `template<size_t _Nb> size_t std::bitset<_Nb>::__Find_next (size_t
__prev) const [inline, inherited]`

Finds the index of the next "on" bit after *prev*.

Returns

The index of the next bit set, or `size()` if not found.

Parameters

prev Where to start searching.

See also

`_Find_first`

Definition at line 1314 of file `bitset`.

2.2.2.5 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>
>::_Unchecked_flip (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 986 of file `bitset`.

2.2.2.6 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>
>::_Unchecked_reset (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 979 of file `bitset`.

2.2.2.7 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>
>::_Unchecked_set (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 962 of file `bitset`.

2.2.2.8 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>
>::_Unchecked_set (size_t __pos, int __val) [inline,
inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 969 of file `bitset`.

2.2.2.9 `template<size_t _Nb> bool std::bitset< _Nb >::_Unchecked_test (`
`size_t __pos) const [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 993 of file `bitset`.

2.2.2.10 `template<class _Operation1 , class _Operation2 >`
`unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1`
`(const _Operation1 & __fn1, const _Operation2 & __fn2)`
`[inline]`

An [SGI extension](#) .

Definition at line 144 of file `ext/functional`.

2.2.2.11 `template<class _Operation1 , class _Operation2 , class _Operation3`
`> binary_compose<_Operation1, _Operation2, _Operation3>`
`__gnu_cxx::compose2 (const _Operation1 & __fn1, const`
`_Operation2 & __fn2, const _Operation3 & __fn3) [inline]`

An [SGI extension](#) .

Definition at line 171 of file `ext/functional`.

2.2.2.12 `template<class _Result > constant_void_fun<_Result>`
`__gnu_cxx::constant0 (const _Result & __val) [inline]`

An [SGI extension](#) .

Definition at line 325 of file `ext/functional`.

2.2.2.13 `template<class _Result > constant_unary_fun<_Result, _Result>`
`__gnu_cxx::constant1 (const _Result & __val) [inline]`

An [SGI extension](#) .

Definition at line 331 of file `ext/functional`.

2.2.2.14 `template<class _Result > constant_binary_fun<_Result, _-`
`Result, _Result> __gnu_cxx::constant2 (const _Result & __val)`
`[inline]`

An [SGI extension](#) .

Definition at line 337 of file ext/functional.

2.2.2.15 `template<typename _InputIterator , typename _Size , typename
_OutputIterator > pair<_InputIterator, _OutputIterator>
__gnu_cxx::copy_n (_InputIterator __first, _Size __count,
_OutputIterator __result) [inline]`

Copies the range [first,first+count) into [result,result+count).

Parameters

first An input iterator.

count The number of elements to copy.

result An output iterator.

Returns

A `std::pair` composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 119 of file ext/algorithm.

References `std::__iterator_category()`.

2.2.2.16 `template<typename _InputIterator , typename _Distance > void
__gnu_cxx::distance (_InputIterator __first, _InputIterator __last,
_Distance & __n) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 103 of file ext/iterator.

References `std::__iterator_category()`.

Referenced by `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::list<_Tp, _Alloc >::size()`.

2.2.2.17 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::plus<_Tp >) [inline]`

An [SGI extension](#) .

Definition at line 86 of file ext/functional.

2.2.2.18 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::multiplies<_Tp >) [inline]`

An [SGI extension](#) .

Definition at line 92 of file ext/functional.

2.2.2.19 `template<typename _ForwardIter , typename _Tp > void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 132 of file ext/numeric.

2.2.2.20 `template<typename _RandomAccessIterator , typename _StrictWeakOrdering > bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 453 of file ext/algorithm.

2.2.2.21 `template<typename _RandomAccessIterator > bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 434 of file ext/algorithm.

2.2.2.22 `template<typename _ForwardIterator > bool __gnu_cxx::is_sorted (`
`_FForwardIterator __first, _ForwardIterator __last)`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 478 of file ext/algorithm.

2.2.2.23 `template<typename _ForwardIterator , typename`
`_StrictWeakOrdering > bool __gnu_cxx::is_sorted (`
`_FForwardIterator __first, _ForwardIterator __last,`
`_StrictWeakOrdering __comp)`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 503 of file ext/algorithm.

2.2.2.24 `template<typename _InputIterator1 , typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1`
`__first1, _InputIterator1 __last1, _InputIterator2 __first2,`
`_InputIterator2 __last2)`

memcmp on steroids.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

last2 An input iterator.

Returns

An int, as with `memcmp`.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 200 of file `ext/algorithm`.

2.2.2.25 `template<typename _Tp , typename _Integer > _Tp
__gnu_cxx::power (_Tp __x, _Integer __n) [inline]`

This is an SGI extension.

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 121 of file `ext/numeric`.

2.2.2.26 `template<typename _Tp , typename _Integer , typename
_MonoidOperation > _Tp __gnu_cxx::power (_Tp __x, _Integer
__n, _MonoidOperation __monoid_op) [inline]`

This is an SGI extension.

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 111 of file `ext/numeric`.

2.2.2.27 `template<typename _InputIterator , typename
_RandomAccessIterator , typename _RandomNumberGenerator >
_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator
__first, _InputIterator __last, _RandomAccessIterator __out_first,
_RandomAccessIterator __out_last, _RandomNumberGenerator &
__rand) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 410 of file ext/algorithm.

```
2.2.2.28  template<typename _InputIterator , typename
            _RandomAccessIterator > _RandomAccessIterator
            __gnu_cxx::random_sample ( _InputIterator __first, _InputIterator
            __last, _RandomAccessIterator __out_first, _RandomAccessIterator
            __out_last ) [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 387 of file ext/algorithm.

```
2.2.2.29  template<typename _ForwardIterator , typename _OutputIterator ,
            typename _Distance , typename _RandomNumberGenerator >
            _OutputIterator __gnu_cxx::random_sample_n ( _ForwardIterator
            __first, _ForwardIterator __last, _OutputIterator __out, const
            _Distance __n, _RandomNumberGenerator & __rand )
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 300 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.30  template<typename _ForwardIterator , typename _OutputIterator ,
            typename _Distance > _OutputIterator __gnu_cxx::random_sample_n
            ( _ForwardIterator __first, _ForwardIterator __last,
            _OutputIterator __out, const _Distance __n )
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 266 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.31  template<typename _InputIter, typename _Size,
              typename _ForwardIter > pair<_InputIter, _ForwardIter>
              __gnu_cxx::uninitialized_copy_n( _InputIter __first, _Size __count,
              _ForwardIter __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

`result + (first - last)`

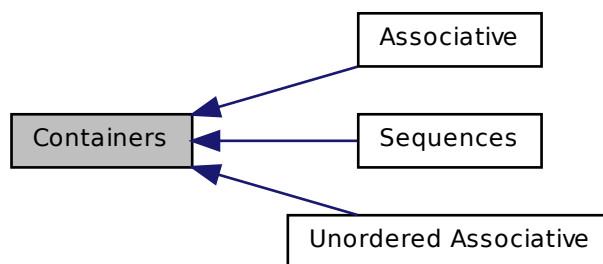
Like `copy()`, but does not require an initialized output range.

Definition at line 121 of file ext/memory.

References `std::__iterator_category()`.

2.3 Containers

Collaboration diagram for Containers:



Classes

- class `std::bitset<_Nb>`

The bitset class represents a fixed-size sequence of bits.

Modules

- [Sequences](#)
- [Associative](#)
- [Unordered Associative](#)

2.3.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in [tables](#).

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

2.4 Sequences

Collaboration diagram for Sequences:



Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
Managing sequences of characters and character-like objects.
- class `std::deque<_Tp, _Alloc>`
A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.
- class `std::forward_list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class `std::list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

- class `std::priority_queue< _Tp, _Sequence, _Compare >`

A standard container automatically sorting its contents.

- class `std::queue< _Tp, _Sequence >`

A standard container giving FIFO behavior.

- class `std::stack< _Tp, _Sequence >`

A standard container giving FILO behavior.

- struct `std::tr1::array< _Tp, _Nm >`

A standard container for storing a fixed size sequence of elements.

- class `std::vector< _Tp, _Alloc >`

A standard container which offers fixed time access to individual elements in any order.

- class `std::vector< bool, _Alloc >`

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

2.4.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

2.5 Associative

Collaboration diagram for Associative:



Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multiset< _Key, _Compare, _Alloc >`
A standard container made up of elements, which can be retrieved in logarithmic time.
- class `std::set< _Key, _Compare, _Alloc >`
A standard container made up of unique keys, which can be retrieved in logarithmic time.

2.5.1 Detailed Description

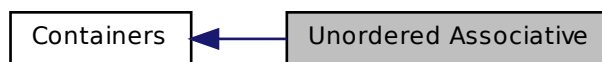
Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

2.6 Unordered Associative

Collaboration diagram for Unordered Associative:



Classes

- class `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

2.6.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

2.7 Diagnostics

Collaboration diagram for Diagnostics:



Modules

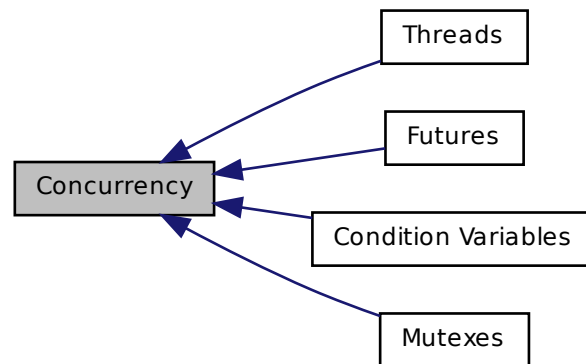
- [Exceptions](#)

2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.8 Concurrency

Collaboration diagram for Concurrency:



Modules

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

2.8.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

2.9 Exceptions

Collaboration diagram for Exceptions:



Classes

- class [__cxxabiv1::__forced_unwind](#)
*Thrown as part of forced unwinding.
 A magic placeholder class that can be caught by reference to recognize forced unwinding.*
- struct [__gnu_cxx::forced_error](#)
Thrown by exception safety machinery.
- class [__gnu_cxx::recursive_init_error](#)
*Exception thrown by `__cxa_guard_acquire`.
 6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class [std::__exception_ptr::exception_ptr](#)
An opaque pointer to an arbitrary exception.
- class [std::bad_alloc](#)
*Exception possibly thrown by `new`.
[bad_alloc](#) (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*
- class [std::bad_cast](#)
*Thrown during incorrect typecasting.
 If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class [std::bad_exception](#)
- class [std::bad_function_call](#)

Exception class thrown when class template function's operator() is called with an empty target.

- class [std::bad_typeid](#)

Thrown when a NULL pointer in a typeid expression is used.

- class [std::domain_error](#)
- class [std::exception](#)

Base class for all library exceptions.

- class [std::future_error](#)

Exception type thrown by futures.

- class [std::invalid_argument](#)
- class [std::ios_base::failure](#)

*These are thrown to indicate problems with io.
27.4.2.1.1 Class [ios_base::failure](#).*

- class [std::length_error](#)
- class [std::logic_error](#)

One of two subclasses of exception.

- class [std::nested_exception](#)

Exception class with exception_ptr data member.

- class [std::out_of_range](#)
- class [std::overflow_error](#)
- class [std::range_error](#)
- class [std::regex_error](#)

*A regular expression exception class.
The regular expression library throws objects of this class on error.*

- class [std::runtime_error](#)

One of two subclasses of exception.

- class [std::system_error](#)

Thrown to indicate error code of underlying system.

- class [std::tr1::bad_weak_ptr](#)

Exception possibly thrown by [shared_ptr](#).

- class [std::underflow_error](#)

Typedefs

- typedef void(* [std::terminate_handler](#))()
- typedef void(* [std::unexpected_handler](#))()

Functions

- template<typename _Ex >
const nested_exception * [std::__get_nested_exception](#) (const _Ex &__ex)
- template<typename _Ex >
void [std::__throw_with_nested](#) (_Ex &&, const nested_exception * = 0) [__attribute__\(\(__noreturn__\)\)](#)
- template<typename _Ex >
void [std::__throw_with_nested](#) (_Ex &&, ...) [__attribute__\(\(__noreturn__\)\)](#)
- void [__gnu_cxx::__verbose_terminate_handler](#) ()
- template<typename _Ex >
exception_ptr [std::copy_exception](#) (_Ex __ex) throw ()
- exception_ptr [std::current_exception](#) () throw ()
- template<typename _Ex >
exception_ptr [std::make_exception_ptr](#) (_Ex __ex) throw ()
- void [std::rethrow_exception](#) (exception_ptr) [__attribute__\(\(__noreturn__\)\)](#)
- void [std::rethrow_if_nested](#) (const nested_exception &__ex)
- template<typename _Ex >
void [std::rethrow_if_nested](#) (const _Ex &__ex)
- terminate_handler [std::set_terminate](#) (terminate_handler) throw ()
- unexpected_handler [std::set_unexpected](#) (unexpected_handler) throw ()
- void [std::terminate](#) () [__attribute__\(\(__noreturn__\)\)](#) throw ()
- template<typename _Ex >
void [std::throw_with_nested](#) (_Ex __ex)
- bool [std::uncaught_exception](#) () [__attribute__\(\(__pure__\)\)](#) throw ()
- void [std::unexpected](#) () [__attribute__\(\(__noreturn__\)\)](#)

2.9.1 Detailed Description

Classes and functions for reporting errors via exception classes.

2.9.2 Typedef Documentation

2.9.2.1 typedef void(* [std::terminate_handler](#))()

If you write a replacement terminate handler, it must be of this type.

Definition at line 88 of file exception.

2.9.2.2 `typedef void(* std::unexpected_handler)()`

If you write a replacement unexpected handler, it must be of this type.

Definition at line 91 of file `exception`.

2.9.3 Function Documentation

2.9.3.1 `void __gnu_cxx::__verbose_terminate_handler ()`

A replacement for the standard `terminate_handler` which prints more information about the terminating exception (if any) on `stderr`.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06>

In 3.4 and later, this is on by default.

2.9.3.2 `template<typename _Ex > exception_ptr std::copy_exception (_Ex __ex) throw ()`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 162 of file `exception_ptr.h`.

References `std::current_exception()`.

2.9.3.3 `exception_ptr std::current_exception () throw ()`

Obtain an `exception_ptr` to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by `std::copy_exception()`.

2.9.3.4 `template<typename _Ex > exception_ptr std::make_exception_ptr (_Ex __ex) throw ()`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 181 of file `exception_ptr.h`.

2.9.3.5 void std::rethrow_exception (exception_ptr)

Throw the object pointed to by the exception_ptr.

Referenced by std::__basic_future< _Res & >::_M_get_result().

**2.9.3.6 void std::rethrow_if_nested (const nested_exception & __ex)
[inline]**

Overload, See N2619.

Definition at line 156 of file nested_exception.h.

**2.9.3.7 template<typename _Ex > void std::rethrow_if_nested (const _Ex &
__ex) [inline]**

If __ex is derived from [nested_exception](#), __ex.rethrow_nested().

Definition at line 148 of file nested_exception.h.

2.9.3.8 terminate_handler std::set_terminate (terminate_handler) throw ()

Takes a new handler function as an argument, returns the old function.

**2.9.3.9 unexpected_handler std::set_unexpected (unexpected_handler)
throw ()**

Takes a new handler function as an argument, returns the old function.

2.9.3.10 void std::terminate () throw ()

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

**2.9.3.11 template<typename _Ex > void std::throw_with_nested (_Ex __ex)
[inline]**

If __ex is derived from [nested_exception](#), __ex. // Else, an implementation-defined object derived from both.

Definition at line 138 of file nested_exception.h.

2.9.3.12 `bool std::uncaught_exception () throw ()`

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).'

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::~~sentry()`.

2.9.3.13 `void std::unexpected ()`

The runtime will call this function if an exception is thrown which violates the function's exception specification.

2.10 Time

Collaboration diagram for Time:



Namespaces

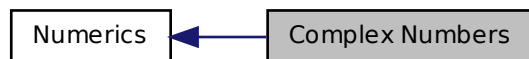
- namespace `std::chrono`

2.10.1 Detailed Description

Classes and functions for time.

2.11 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- struct `std::complex< _Tp >`

Functions

- `std::complex< float >::complex` (const complex< double > &)
- `std::complex< float >::complex` (const complex< long double > &)
- `std::complex< double >::complex` (const complex< long double > &)
- `template<typename _Tp >`
`_Tp std::__complex_abs` (const complex< _Tp > &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_acos` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_acosh` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`_Tp std::__complex_arg` (const complex< _Tp > &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_asin` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_asinh` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_atan` (const `std::complex< _Tp >` &__z)

- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_atanh (const std::complex< _Tp >`
`&__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const`
`complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::arg (_Tp __x)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::conj (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::imag (_Tp)`
- `template<typename _Tp >`
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::norm (_Tp __x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator*= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator*= (const _Tp &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator-= (const complex< _Up > &)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator/= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator/= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`
`_Traits > &__is, complex< _Tp > &__x)`

- `template<typename _Tp >`
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`_Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::real (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tanh (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

2.11.1 Detailed Description

Classes and functions for complex numbers.

2.11.2 Function Documentation

2.11.2.1 `template<typename _Tp> _Tp std::abs (const complex< _Tp> & __z) [inline]`

Return magnitude of z .

Definition at line 594 of file `complex`.

Referenced by `std::tr1::fabs()`, `std::binomial_distribution< _IntType>::operator()()`, and `std::poisson_distribution< _IntType>::operator()()`.

2.11.2.2 `template<typename _Tp> std::complex< _Tp> std::tr1::acos (const std::complex< _Tp> & __z) [inline]`

`acos(__z)` [8.1.2].

Definition at line 86 of file `tr1_impl/complex`.

2.11.2.3 `template<typename _Tp> std::complex< _Tp> std::tr1::acosh (const std::complex< _Tp> & __z) [inline]`

`acosh(__z)` [8.1.5].

Definition at line 205 of file `tr1_impl/complex`.

2.11.2.4 `template<typename _Tp> __gnu_cxx::__promote< _Tp>::__type std::tr1::arg (_Tp __x) [inline]`

Additional overloads [8.1.9].

Definition at line 311 of file `tr1_impl/complex`.

References `std::arg()`.

2.11.2.5 `template<typename _Tp> _Tp std::arg (const complex< _Tp> & __z) [inline]`

Return phase angle of z .

Definition at line 621 of file `complex`.

Referenced by `std::tr1::arg()`.

2.11.2.6 `template<typename _Tp > std::complex< _Tp > std::tr1::asin (`
`const std::complex< _Tp > & __z) [inline]`

`asin(__z)` [8.1.3].

Definition at line 122 of file `tr1_impl/complex`.

2.11.2.7 `template<typename _Tp > std::complex< _Tp > std::tr1::asinh (`
`const std::complex< _Tp > & __z) [inline]`

`asinh(__z)` [8.1.6].

Definition at line 244 of file `tr1_impl/complex`.

2.11.2.8 `template<typename _Tp > std::complex< _Tp > std::tr1::atan (`
`const std::complex< _Tp > & __z) [inline]`

`atan(__z)` [8.1.4].

Definition at line 166 of file `tr1_impl/complex`.

2.11.2.9 `template<typename _Tp > std::complex< _Tp > std::tr1::atanh (`
`const std::complex< _Tp > & __z) [inline]`

`atanh(__z)` [8.1.7].

Definition at line 288 of file `tr1_impl/complex`.

2.11.2.10 `template<typename _Tp > complex< _Tp > std::conj (const`
`complex< _Tp > & __z) [inline]`

Return complex conjugate of z .

Definition at line 667 of file `complex`.

2.11.2.11 `template<typename _Tp > complex< _Tp > std::cos (const`
`complex< _Tp > & __z) [inline]`

Return complex cosine of z .

Definition at line 699 of file `complex`.

Referenced by `std::polar()`.

2.11.2.12 `template<typename _Tp > complex< _Tp > std::cosh (const
complex< _Tp > & __z) [inline]`

Return complex hyperbolic cosine of z .

Definition at line 729 of file `complex`.

2.11.2.13 `template<typename _Tp > complex< _Tp > std::exp (const
complex< _Tp > & __z) [inline]`

Return complex base e exponential of z .

Definition at line 755 of file `complex`.

Referenced by `std::pow()`.

2.11.2.14 `template<typename _Tp > _Tp std::tr1::fabs (const std::complex<
_Tp > & __z) [inline]`

`fabs(__z)` [8.1.8].

Definition at line 301 of file `tr1_impl/complex`.

References `std::abs()`.

2.11.2.15 `template<typename _Tp > complex< _Tp > std::log (const
complex< _Tp > & __z) [inline]`

Return complex natural logarithm of z .

Definition at line 782 of file `complex`.

Referenced by `std::generate_canonical()`, `std::log10()`, `std::gamma_distribution< _RealType >::operator()`, `std::normal_distribution< _RealType >::operator()`, `std::binomial_distribution< _IntType >::operator()`, `std::poisson_distribution< _IntType >::operator()`, `std::independent_bits_engine< _RandomNumberEngine, _w, _UIntType >::operator()`, and `std::pow()`.

2.11.2.16 `template<typename _Tp > complex< _Tp > std::log10 (const
complex< _Tp > & __z) [inline]`

Return complex base 10 logarithm of z .

Definition at line 787 of file `complex`.

References `std::log()`.

2.11.2.17 `template<typename _Tp > _Tp std::norm (const complex< _Tp > & __z) [inline]`

Return z magnitude squared.

Definition at line 654 of file complex.

Referenced by `std::complex< _Tp >::operator/=()`.

2.11.2.18 `template<typename _Tp > bool std::operator!=(const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return false if x is equal to y .

Definition at line 469 of file complex.

2.11.2.19 `template<typename _Tp > bool std::operator!=(const complex< _Tp > & __x, const _Tp & __y) [inline]`

Return false if x is equal to y .

Definition at line 474 of file complex.

2.11.2.20 `template<typename _Tp > bool std::operator!=(const _Tp & __x, const complex< _Tp > & __y) [inline]`

Return false if x is equal to y .

Definition at line 479 of file complex.

2.11.2.21 `template<typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return new complex value x times y .

Definition at line 379 of file complex.

2.11.2.22 `template<typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & __x, const _Tp & __y) [inline]`

Return new complex value x times y .

Definition at line 388 of file complex.

2.11.2.23 `template<typename _Tp > complex<_Tp> std::operator* (const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x times y .

Definition at line 397 of file complex.

2.11.2.24 `template<typename _Tp > complex<_Tp> & std::complex<_Tp
>::operator*=(const _Tp & __t) [inherited]`

Multiply this complex number by t .

Definition at line 238 of file complex.

2.11.2.25 `template<typename _Tp > template<typename _Up > complex<
_Tp> & std::complex<_Tp>::operator*=(const complex<_Up>
& __z) [inherited]`

Multiply this complex number by z .

Definition at line 292 of file complex.

2.11.2.26 `template<typename _Tp > complex<_Tp> std::operator+ (const
complex<_Tp> & __x) [inline]`

Return x .

Definition at line 438 of file complex.

2.11.2.27 `template<typename _Tp > complex<_Tp> std::operator+ (const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x plus y .

Definition at line 337 of file complex.

2.11.2.28 `template<typename _Tp > complex<_Tp> std::operator+ (const
complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x plus y .

Definition at line 328 of file complex.

2.11.2.29 `template<typename _Tp> complex<_Tp> std::operator+ (`
 `const complex<_Tp> & __x, const complex<_Tp> & __y)`
 `[inline]`

Return new complex value x plus y .

Definition at line 319 of file complex.

2.11.2.30 `template<typename _Tp> template<typename _Up> complex<`
 `_Tp> & std::complex<_Tp>::operator+=(const complex<_Up>`
 `& __z) [inherited]`

Add z to this complex number.

Definition at line 269 of file complex.

2.11.2.31 `template<typename _Tp> complex<_Tp> std::operator- (const`
 `complex<_Tp> & __x) [inline]`

Return complex negation of x .

Definition at line 444 of file complex.

2.11.2.32 `template<typename _Tp> complex<_Tp> std::operator- (`
 `const complex<_Tp> & __x, const complex<_Tp> & __y)`
 `[inline]`

Return new complex value x minus y .

Definition at line 349 of file complex.

2.11.2.33 `template<typename _Tp> complex<_Tp> std::operator- (const`
 `complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x minus y .

Definition at line 358 of file complex.

2.11.2.34 `template<typename _Tp> complex<_Tp> std::operator- (const`
 `_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x minus y .

Definition at line 367 of file complex.

2.11.2.35 `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator-= (const complex<_Up > & __z) [inherited]`

Subtract z from this complex number.

Definition at line 280 of file `complex`.

2.11.2.36 `template<typename _Tp > complex<_Tp> std::operator/ (const complex<_Tp > & __x, const complex<_Tp > & __y) [inline]`

Return new complex value x divided by y .

Definition at line 409 of file `complex`.

2.11.2.37 `template<typename _Tp > complex<_Tp> std::operator/ (const complex<_Tp > & __x, const _Tp & __y) [inline]`

Return new complex value x divided by y .

Definition at line 418 of file `complex`.

2.11.2.38 `template<typename _Tp > complex<_Tp> std::operator/ (const _Tp & __x, const complex<_Tp > & __y) [inline]`

Return new complex value x divided by y .

Definition at line 427 of file `complex`.

2.11.2.39 `template<typename _Tp > complex<_Tp > & std::complex<_Tp >::operator/= (const _Tp & __t) [inherited]`

Divide this complex number by t .

Definition at line 248 of file `complex`.

2.11.2.40 `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator/= (const complex<_Up > & __z) [inherited]`

Divide this complex number by z .

Definition at line 305 of file `complex`.

References `std::norm()`.

2.11.2.41 `template<typename _Tp, typename _CharT, class _Traits
> basic_ostream<_CharT, _Traits>& std::operator<< (
basic_ostream<_CharT, _Traits> & __os, const complex<_Tp>
& __x)`

Insertion operator for complex values.

Definition at line 519 of file complex.

References `std::ios_base::flags()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::ios_base::precision()`, and `std::basic_ostringstream<_CharT, _Traits, _Alloc>::str()`.

2.11.2.42 `template<typename _Tp> complex<_Tp> & std::complex<_Tp
>::operator=(const _Tp & __t) [inherited]`

Assign this complex number to scalar t .

Definition at line 228 of file complex.

2.11.2.43 `template<typename _Tp> template<typename _Up> complex<
_Tp> & std::complex<_Tp>::operator=(const complex<_Up>
& __z) [inherited]`

Assign this complex number to complex z .

Definition at line 258 of file complex.

2.11.2.44 `template<typename _Tp> bool std::operator==(const complex<
_Tp> & __x, const _Tp & __y) [inline]`

Return true if x is equal to y .

Definition at line 456 of file complex.

2.11.2.45 `template<typename _Tp> bool std::operator==(const complex<
_Tp> & __x, const complex<_Tp> & __y) [inline]`

Return true if x is equal to y .

Definition at line 451 of file complex.

2.11.2.46 `template<typename _Tp > bool std::operator==(const _Tp & __x,
const complex< _Tp > & __y) [inline]`

Return true if x is equal to y .

Definition at line 461 of file `complex`.

2.11.2.47 `template<typename _Tp , typename _CharT , class _Traits
> basic_istream< _CharT, _Traits>& std::operator>> (
basic_istream< _CharT, _Traits > & __is, complex< _Tp > & __x)`

Extraction operator for complex values.

Definition at line 486 of file `complex`.

References `std::ios_base::failbit`, `std::basic_istream< _CharT, _Traits >::putback()`,
and `std::basic_ios< _CharT, _Traits >::setstate()`.

2.11.2.48 `template<typename _Tp > complex< _Tp > std::polar (const _Tp
& __rho, const _Tp & __theta = 0) [inline]`

Return complex with magnitude ρ and angle θ .

Definition at line 662 of file `complex`.

References `std::cos()`, and `std::sin()`.

Referenced by `std::pow()`.

2.11.2.49 `template<typename _Tp > complex< _Tp > std::pow (const _Tp &
__x, const complex< _Tp > & __y) [inline]`

Return x to the y 'th power.

Definition at line 1009 of file `complex`.

References `std::log()`, `std::polar()`, and `std::pow()`.

2.11.2.50 `template<typename _Tp > complex< _Tp > std::pow (const
complex< _Tp > & __x, const _Tp & __y)`

Return x to the y 'th power.

Definition at line 964 of file `complex`.

References `std::exp()`, `std::log()`, and `std::polar()`.

Referenced by `std::gamma_distribution< _RealType >::operator()()`, and `std::pow()`.

2.11.2.51 `template<typename _Tp> complex<_Tp> std::pow (const
complex<_Tp> & __x, const complex<_Tp> & __y)
[inline]`

Return x to the y 'th power.

Definition at line 1003 of file `complex`.

2.11.2.52 `template<typename _Tp> complex<_Tp> std::sin (const
complex<_Tp> & __z) [inline]`

Return complex sine of z .

Definition at line 817 of file `complex`.

Referenced by `std::polar()`.

2.11.2.53 `template<typename _Tp> complex<_Tp> std::sinh (const
complex<_Tp> & __z) [inline]`

Return complex hyperbolic sine of z .

Definition at line 847 of file `complex`.

2.11.2.54 `template<typename _Tp> complex<_Tp> std::sqrt (const
complex<_Tp> & __z) [inline]`

Return complex square root of z .

Definition at line 891 of file `complex`.

Referenced by `std::normal_distribution<_RealType>::operator()()`, and
`std::student_t_distribution<_RealType>::operator()()`.

2.11.2.55 `template<typename _Tp> complex<_Tp> std::tan (const
complex<_Tp> & __z) [inline]`

Return complex tangent of z .

Definition at line 918 of file `complex`.

2.11.2.56 `template<typename _Tp> complex<_Tp> std::tanh (const
complex<_Tp> & __z) [inline]`

Return complex hyperbolic tangent of z .

Definition at line 946 of file complex.

2.12 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class [std::condition_variable](#)
condition_variable
- class [std::condition_variable_any](#)
condition_variable_any

Enumerations

- enum [std::cv_status](#) { **no_timeout**, **timeout** }

2.12.1 Detailed Description

Classes for [condition_variable](#) support.

2.12.2 Enumeration Type Documentation

2.12.2.1 enum std::cv_status

`cv_status`

Definition at line 54 of file `condition_variable`.

2.13 Futures

Collaboration diagram for Futures:



Classes

- class `std::__basic_future<_Res>`
Common implementation for future and `shared_future`.
- struct `std::__future_base`
Base class and enclosing scope.
- struct `std::__future_base::Result<_Res &>`
Partial specialization for reference types.
- struct `std::__future_base::Result<void>`
Explicit specialization for void.
- class `std::future<_Res>`
Primary template for future.
- class `std::future<_Res &>`
Partial specialization for `future<R&>`
- class `std::future<void>`
Explicit specialization for `future<void>`
- class `std::future_error`
Exception type thrown by futures.
- class `std::packaged_task<_Res(_ArgTypes...)>`
`packaged_task`

- class `std::promise< _Res >`
Primary template for `promise`.
- class `std::promise< _Res & >`
Partial specialization for `promise<R&>`
- class `std::promise< void >`
Explicit specialization for `promise<void>`
- class `std::shared_future< _Res >`
Primary template for `shared_future`.
- class `std::shared_future< _Res & >`
Partial specialization for `shared_future<R&>`
- class `std::shared_future< void >`
Explicit specialization for `shared_future<void>`

Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }

Functions

- `std::__basic_future::__basic_future` (const `shared_future< _Res > &`)
- `std::__basic_future::__basic_future` (`shared_future< _Res > &&`)
- `std::__basic_future::__basic_future` (`future< _Res > &&`)
- static `_Setter< void, void > std::__future_base::State::__setter` (`promise< void > *__prom`)
- `template<typename _Fn, typename... _Args>`
`enable_if<!is_same< typename decay< _Fn >::type, launch >::value, future<`
`decltype(std::declval< _Fn >)(std::declval< _Args >)...)> >::type std::async`
`(_Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`
`future< typename result_of< _Fn(_Args...)>::type > std::async` (`launch __-`
`policy, _Fn &&__fn, _Args &&...__args)`
- error_code `std::make_error_code` (`future_errc __errc`)
- error_condition `std::make_error_condition` (`future_errc __errc`)

- void **std::promise**< **void** >::**set_value** ()
- template<typename _Res , typename... _ArgTypes>
void **std::swap** (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)
- template<typename _Res >
void **std::swap** (promise< _Res > &__x, promise< _Res > &__y)

Variables

- const error_category *const [std::future_category](#)

2.13.1 Detailed Description

Classes for futures support.

2.13.2 Enumeration Type Documentation

2.13.2.1 enum std::future_errc

Error code for futures.

Definition at line 59 of file future.

2.13.3 Variable Documentation

2.13.3.1 const error_category* const std::future_category

Points to a statically-allocated object derived from [error_category](#).

2.14 I/O

Classes

- class [__gnu_cxx::stdio_filebuf](#)< _CharT, _Traits >
*Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.*
- class [__gnu_cxx::stdio_sync_filebuf](#)< _CharT, _Traits >

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE's. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

- class `std::basic_filebuf<_CharT, _Traits>`

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.

- class `std::basic_fstream<_CharT, _Traits>`

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ifstream<_CharT, _Traits>`

Controlling input for files.

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ios<_CharT, _Traits>`

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

- class `std::basic_iostream<_CharT, _Traits>`

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

- class `std::basic_istream<_CharT, _Traits>`

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.

- class `std::basic_istream<_CharT, _Traits, _Alloc>`

Controlling input for `std::string`.

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::basic_ofstream<_CharT, _Traits>`

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ostream< _CharT, _Traits >`

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

- class `std::basic_ostringstream< _CharT, _Traits, _Alloc >`

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::basic_streambuf< _CharT, _Traits >`

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::ios_base`

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Typedefs

- typedef `basic_filebuf< char >` `std::filebuf`
- typedef `basic_fstream< char >` `std::fstream`

- `typedef basic_ifstream< char > std::ifstream`
- `typedef basic_ios< char > std::ios`
- `typedef basic_iostream< char > std::iostream`
- `typedef basic_istream< char > std::istream`
- `typedef basic_istreamstream< char > std::istreamstream`
- `typedef basic_ofstream< char > std::ofstream`
- `typedef basic_ostream< char > std::ostream`
- `typedef basic_ostreamstream< char > std::ostreamstream`
- `typedef basic_streambuf< char > std::streambuf`
- `typedef basic_stringbuf< char > std::stringbuf`
- `typedef basic_stringstream< char > std::stringstream`
- `typedef basic_filebuf< wchar_t > std::wfilebuf`
- `typedef basic_fstream< wchar_t > std::wfstream`
- `typedef basic_ifstream< wchar_t > std::wifstream`
- `typedef basic_ios< wchar_t > std::wios`
- `typedef basic_iostream< wchar_t > std::wiostream`
- `typedef basic_istream< wchar_t > std::wistream`
- `typedef basic_istreamstream< wchar_t > std::wistreamstream`
- `typedef basic_ofstream< wchar_t > std::wofstream`
- `typedef basic_ostream< wchar_t > std::wostream`
- `typedef basic_ostreamstream< wchar_t > std::wostreamstream`
- `typedef basic_streambuf< wchar_t > std::wstreambuf`
- `typedef basic_stringbuf< wchar_t > std::wstringbuf`
- `typedef basic_stringstream< wchar_t > std::wstringstream`

2.14.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt>

2.14.2 Typedef Documentation

2.14.2.1 `typedef basic_filebuf<char> std::filebuf`

One of the [I/O](#) typedefs.

Definition at line 137 of file iosfwd.

2.14.2.2 `typedef basic_fstream<char> std::fstream`

One of the [I/O](#) typedefs.

Definition at line 140 of file iosfwd.

2.14.2.3 `typedef basic_ifstream<char> std::ifstream`

One of the [I/O](#) typedefs.

Definition at line 138 of file iosfwd.

2.14.2.4 `typedef basic_ios<char> std::ios`

One of the [I/O](#) typedefs.

Definition at line 123 of file iosfwd.

2.14.2.5 `typedef basic_iostream<char> std::iostream`

One of the [I/O](#) typedefs.

Definition at line 132 of file iosfwd.

2.14.2.6 `typedef basic_istream<char> std::istream`

One of the [I/O](#) typedefs.

Definition at line 130 of file iosfwd.

2.14.2.7 `typedef basic_istreamstream<char> std::istreamstream`

One of the [I/O](#) typedefs.

Definition at line 134 of file iosfwd.

2.14.2.8 typedef basic_ofstream<char> std::ofstream

One of the [I/O](#) typedefs.

Definition at line 139 of file iosfwd.

2.14.2.9 typedef basic_ostream<char> std::ostream

One of the [I/O](#) typedefs.

Definition at line 131 of file iosfwd.

2.14.2.10 typedef basic_ostringstream<char> std::ostringstream

One of the [I/O](#) typedefs.

Definition at line 135 of file iosfwd.

2.14.2.11 typedef basic_streambuf<char> std::streambuf

One of the [I/O](#) typedefs.

Definition at line 129 of file iosfwd.

2.14.2.12 typedef basic_stringbuf<char> std::stringbuf

One of the [I/O](#) typedefs.

Definition at line 133 of file iosfwd.

2.14.2.13 typedef basic_stringstream<char> std::stringstream

One of the [I/O](#) typedefs.

Definition at line 136 of file iosfwd.

2.14.2.14 typedef basic_filebuf<wchar_t> std::wfilebuf

One of the [I/O](#) typedefs.

Definition at line 152 of file iosfwd.

2.14.2.15 typedef basic_fstream<wchar_t> std::wfstream

One of the [I/O](#) typedefs.

Definition at line 155 of file iosfwd.

2.14.2.16 typedef basic_ifstream<wchar_t> std::wifstream

One of the [I/O](#) typedefs.

Definition at line 153 of file iosfwd.

2.14.2.17 typedef basic_ios<wchar_t> std::wios

One of the [I/O](#) typedefs.

Definition at line 143 of file iosfwd.

2.14.2.18 typedef basic_iostream<wchar_t> std::wiostream

One of the [I/O](#) typedefs.

Definition at line 147 of file iosfwd.

2.14.2.19 typedef basic_istream<wchar_t> std::wistream

One of the [I/O](#) typedefs.

Definition at line 145 of file iosfwd.

2.14.2.20 typedef basic_istreamstream<wchar_t> std::wistreamstream

One of the [I/O](#) typedefs.

Definition at line 149 of file iosfwd.

2.14.2.21 typedef basic_ofstream<wchar_t> std::wofstream

One of the [I/O](#) typedefs.

Definition at line 154 of file iosfwd.

2.14.2.22 typedef basic_ostream<wchar_t> std::wostream

One of the [I/O](#) typedefs.

Definition at line 146 of file iosfwd.

2.14.2.23 typedef basic_ostringstream<wchar_t> std::wostringstream

One of the [I/O](#) typedefs.

Definition at line 150 of file iosfwd.

2.14.2.24 typedef basic_streambuf<wchar_t> std::wstreambuf

One of the [I/O](#) typedefs.

Definition at line 144 of file iosfwd.

2.14.2.25 typedef basic_stringbuf<wchar_t> std::wstringbuf

One of the [I/O](#) typedefs.

Definition at line 148 of file iosfwd.

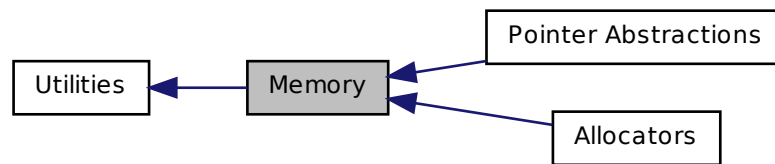
2.14.2.26 typedef basic_stringstream<wchar_t> std::wstringstream

One of the [I/O](#) typedefs.

Definition at line 151 of file iosfwd.

2.15 Memory

Collaboration diagram for Memory:



Modules

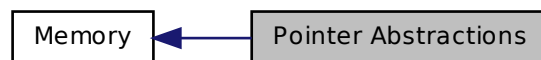
- [Pointer Abstractions](#)
- [Allocators](#)

2.15.1 Detailed Description

Components for memory allocation, deallocation, and management.

2.16 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- struct `std::default_delete< _Tp >`
Primary template, [default_delete](#).
- struct `std::default_delete< _Tp[]>`
Specialization, [default_delete](#).
- class `std::enable_shared_from_this< _Tp >`
Base class allowing use of member function [shared_from_this](#).
- struct `std::hash< shared_ptr< _Tp > >`
[std::hash](#) specialization for [shared_ptr](#).
- struct `std::hash< unique_ptr< _Tp, _Tp_Deleter > >`
[std::hash](#) specialization for [unique_ptr](#).
- struct `std::owner_less< shared_ptr< _Tp > >`
Partial specialization of [owner_less](#) for [shared_ptr](#).
- struct `std::owner_less< weak_ptr< _Tp > >`
Partial specialization of [owner_less](#) for [weak_ptr](#).
- class `std::shared_ptr< _Tp >`
A smart pointer with reference-counted copy semantics.
- class `std::unique_ptr< _Tp, _Tp_Deleter >`
20.7.12.2 [unique_ptr](#) for single objects.
- class `std::unique_ptr< _Tp[], _Tp_Deleter >`
20.7.12.3 [unique_ptr](#) for array objects with a runtime length
- class `std::weak_ptr< _Tp >`
A smart pointer with weak semantics.

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
`shared_ptr< _Tp > std::allocate_shared (_Alloc __a, _Args &&...__args)`
- template<typename _Tp, typename _Tp1 >
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r)`

- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 >`
`&__r)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > std::make_shared (_Args &&...__args)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator!= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator< (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch,`
`_Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator<= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator== (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp_Deleter >`
`void std::swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _-`
`Tp_Deleter > &__y)`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`

2.16.1 Detailed Description

Smart pointers, etc.

2.16.2 Function Documentation

2.16.2.1 `template<typename _Tp , typename _Alloc , typename... _Args>`
`shared_ptr<_Tp> std::allocate_shared (_Alloc __a, _Args &&...
__args) [inline]`

Create an object that is owned by a [shared_ptr](#).

Parameters

`__a` An allocator.

`__args` Arguments for the `_Tp` object's constructor.

Returns

A [shared_ptr](#) that owns the newly created object.

Exceptions

An exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the [shared_ptr](#) and the new object.

Definition at line 495 of file `shared_ptr.h`.

2.16.2.2 `template<typename _Del , typename _Tp , _Lock_policy _Lp>`
`_Del* std::get_deleter (const __shared_ptr< _Tp, _Lp > & __p)`
`[inline]`

2.2.3.10 [shared_ptr](#) `get_deleter` (experimental)

Definition at line 74 of file `shared_ptr.h`.

2.16.2.3 `template<typename _Tp , typename... _Args> shared_ptr<_Tp>`
`std::make_shared (_Args &&... __args) [inline]`

Create an object that is owned by a [shared_ptr](#).

Parameters

`__args` Arguments for the `_Tp` object's constructor.

Returns

A [shared_ptr](#) that owns the newly created object.

Exceptions

[std::bad_alloc](#), or an exception thrown from the constructor of `_Tp`.

Definition at line 510 of file `shared_ptr.h`.

```
2.16.2.4 template<typename _Ch , typename _Tr , typename _Tp
, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>&
std::operator<<( std::basic_ostream<_Ch, _Tr> & __os, const
__shared_ptr<_Tp, _Lp> & __p ) [inline]
```

2.2.3.7 [shared_ptr](#) I/O

Definition at line 64 of file `shared_ptr.h`.

2.17 Mutexes

Collaboration diagram for Mutexes:

**Classes**

- struct [std::adopt_lock_t](#)
Assume the calling thread has already obtained mutex ownership /// and manage it.
- struct [std::defer_lock_t](#)
Do not acquire ownership of the mutex.

- class `std::lock_guard< _Mutex >`
Scoped lock idiom.
- class `std::mutex`
mutex
- struct `std::once_flag`
once_flag
- class `std::recursive_mutex`
recursive_mutex
- class `std::recursive_timed_mutex`
recursive_timed_mutex
- class `std::timed_mutex`
timed_mutex
- struct `std::try_to_lock_t`
Try to acquire ownership of the mutex without blocking.
- class `std::unique_lock< _Mutex >`
unique_lock

Functions

- `mutex & std::__get_once_mutex ()`
- `void std::__once_proxy ()`
- `void std::__set_once_functor_lock_ptr (unique_lock< mutex > *)`
- `template<typename _Callable, typename... _Args>`
`void std::call_once (once_flag &__once, _Callable __f, _Args &&...__args)`
- `template<typename _L1, typename _L2, typename... _L3>`
`void std::lock (_L1 &, _L2 &, _L3 &...)`
- `template<typename _Mutex >`
`void std::swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`
`int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`

Variables

- function< void()> **std::__once_functor**
- const adopt_lock_t **std::adopt_lock**
- const defer_lock_t **std::defer_lock**
- const try_to_lock_t **std::try_to_lock**

2.17.1 Detailed Description

Classes for mutex support.

2.17.2 Function Documentation

2.17.2.1 `template<typename _Callable , typename... _Args> void
std::call_once (once_flag & __once, _Callable __f, _Args &&...
__args)`

call_once

Definition at line 721 of file mutex.

2.17.2.2 `template<typename _L1 , typename _L2 , typename... _L3> void
std::lock (_L1 &, _L2 &, _L3 & ...)`

lock

2.17.2.3 `template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
int std::try_lock (_Lock1 & __l1, _Lock2 & __l2, _Lock3 &... __l3
)`

Generic try_lock.

Parameters

- `__l1` Meets Mutex requirements ([try_lock\(\)](#) may throw).
- `__l2` Meets Mutex requirements ([try_lock\(\)](#) may throw).
- `__l3` Meets Mutex requirements ([try_lock\(\)](#) may throw).

Returns

Returns -1 if all [try_lock\(\)](#) calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls [try_lock\(\)](#) on each argument.

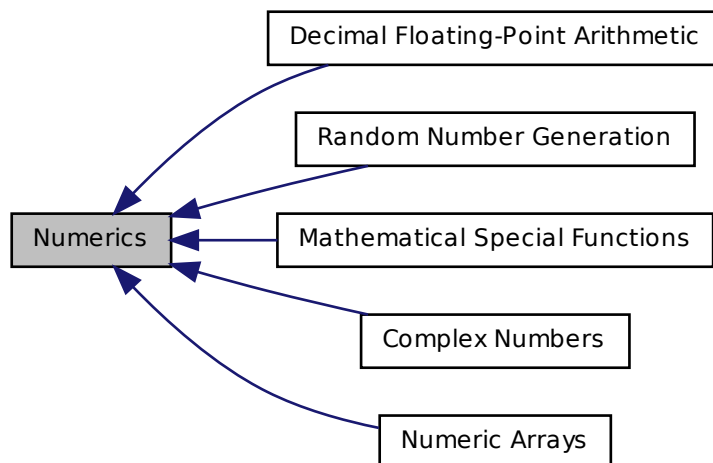
Definition at line 663 of file mutex.

References `std::try_lock()`.

Referenced by `std::try_lock()`.

2.18 Numerics

Collaboration diagram for Numerics:

**Modules**

- [Complex Numbers](#)
- [Numeric Arrays](#)
- [Mathematical Special Functions](#)
- [Decimal Floating-Point Arithmetic](#)
- [Random Number Generation](#)

2.18.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

2.19 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Classes

- struct `std::ratio< _Num, _Den >`
Provides compile-time rational arithmetic.
- struct `std::ratio_add< _R1, _R2 >`
ratio_add
- struct `std::ratio_divide< _R1, _R2 >`
ratio_divide
- struct `std::ratio_equal< _R1, _R2 >`
ratio_equal
- struct `std::ratio_greater< _R1, _R2 >`
ratio_greater
- struct `std::ratio_greater_equal< _R1, _R2 >`
ratio_greater_equal
- struct `std::ratio_less< _R1, _R2 >`

ratio_less

- struct `std::ratio_less_equal<_R1, _R2>`
ratio_less_equal
- struct `std::ratio_multiply<_R1, _R2>`
ratio_multiply
- struct `std::ratio_not_equal<_R1, _R2>`
ratio_not_equal
- struct `std::ratio_subtract<_R1, _R2>`
ratio_subtract

Typedefs

- typedef `ratio< 1, 1000000000000000000 >` **std::atto**
- typedef `ratio< 1, 100 >` **std::centi**
- typedef `ratio< 10, 1 >` **std::deca**
- typedef `ratio< 1, 10 >` **std::deci**
- typedef `ratio< 1000000000000000000, 1 >` **std::exa**
- typedef `ratio< 1, 1000000000000000 >` **std::femto**
- typedef `ratio< 1000000000, 1 >` **std::giga**
- typedef `ratio< 100, 1 >` **std::hecto**
- typedef `ratio< 1000, 1 >` **std::kilo**
- typedef `ratio< 1000000, 1 >` **std::mega**
- typedef `ratio< 1, 1000000 >` **std::micro**
- typedef `ratio< 1, 1000 >` **std::milli**
- typedef `ratio< 1, 1000000000 >` **std::nano**
- typedef `ratio< 1000000000000000000, 1 >` **std::peta**
- typedef `ratio< 1, 1000000000000 >` **std::pico**
- typedef `ratio< 10000000000000, 1 >` **std::tera**

Variables

- static const intmax_t **std::ratio::den**
- static const intmax_t **std::ratio::num**

2.19.1 Detailed Description

Compile time representation of finite rational numbers.

2.20 Threads

Collaboration diagram for Threads:



Classes

- struct `std::hash< thread::id >`
std::hash specialization for thread::id.
- class `std::thread`
thread

Namespaces

- namespace `std::this_thread`

Functions

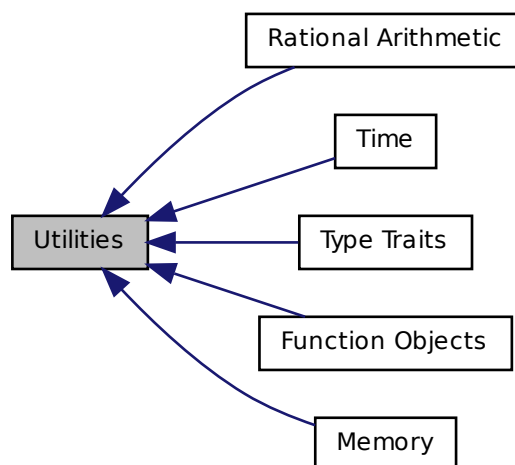
- bool **std::operator!=** (thread::id __x, thread::id __y)
- template<class _CharT, class _Traits >
 basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _-
 CharT, _Traits > &__out, thread::id __id)
- bool **std::operator<=** (thread::id __x, thread::id __y)
- bool **std::operator>** (thread::id __x, thread::id __y)
- bool **std::operator>=** (thread::id __x, thread::id __y)
- void **std::swap** (thread &__x, thread &__y)

2.20.1 Detailed Description

Classes for thread support.

2.21 Utilities

Collaboration diagram for Utilities:



Modules

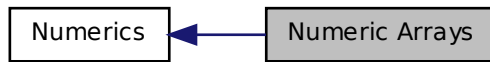
- [Time](#)
- [Memory](#)
- [Rational Arithmetic](#)
- [Type Traits](#)
- [Function Objects](#)

2.21.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

2.22 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class `std::gslice`
Class defining multi-dimensional subset of an array.
- class `std::gslice_array< _Tp >`
Reference to multi-dimensional subset of an array.
- class `std::indirect_array< _Tp >`
Reference to arbitrary subset of an array.
- class `std::mask_array< _Tp >`
Reference to selected subset of an array.
- class `std::slice`
Class defining one-dimensional subset of an array.
- class `std::slice_array< _Tp >`
Reference to one-dimensional subset of an array.
- class `std::valarray< _Tp >`
Smart array designed to support numeric processing.

Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`

- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t, const valarray< size_t > &, const valarray< size_t > &)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array::gslice_array (const gslice_array &)`
- `std::indirect_array::indirect_array (const indirect_array &)`
- `std::mask_array::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t, size_t, size_t)`
- `std::slice_array::slice_array (const slice_array &)`
- `std::valarray::valarray (const mask_array< _Tp > &)`
- `template<class _Dom >`
`std::valarray::valarray (const _Expr< _Dom, _Tp > &_e)`
- `std::valarray::valarray (const indirect_array< _Tp > &)`
- `std::valarray::valarray (initializer_list< _Tp >)`
- `std::valarray::valarray ()`
- `std::valarray::valarray (size_t)`
- `std::valarray::valarray (const _Tp &, size_t)`
- `template<typename _Tp>`
`std::valarray::valarray (const _Tp *__restrict __p, size_t __n)`
- `std::valarray::valarray (const valarray &)`
- `std::valarray::valarray (const slice_array< _Tp > &)`
- `std::valarray::valarray (const gslice_array< _Tp > &)`
- `std::gslice::~gslice ()`
- `_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp func(_Tp)) const`
- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp func(const _Tp &)) const`
- `valarray< _Tp > std::valarray::cshift (int) const`
- `_Tp std::valarray::max () const`
- `_Tp std::valarray::min () const`

- `_UnaryOp< __logical_not >::_Rt std::valarray::operator! () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `void std::gslice_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator%= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator%= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator%= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp`
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator&= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator&= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator&= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array::operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator*= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator*= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator*= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator*= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const _Tp &__t)`

- `_UnaryOp< __unary_plus >::_Rt std::valarray::operator+ () const`
- `template<typename _Tp >
_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _-
_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >
_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _-
_fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const
valarray< _Tp > &__v)`
- `void std::gslice_array::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >
void std::gslice_array::operator+= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator+= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator+= (const valarray< _Tp > &)`
- `template<class _Dom >
valarray< _Tp > & std::valarray::operator+= (const _Expr< _Dom, _Tp >
&)`
- `template<typename _Tp >
_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _-
_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >
_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _-
_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >
&__v, const _Tp &__t)`
- `_UnaryOp< __negate >::_Rt std::valarray::operator- () const`
- `template<typename _Tp >
_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _-
_fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const
valarray< _Tp > &__v)`
- `void std::gslice_array::operator-= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator-= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator-= (const _Tp &)`
- `template<class _Dom >
void std::gslice_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >
valarray< _Tp > & std::valarray::operator-= (const _Expr< _Dom, _Tp >
&)`
- `template<typename _Tp >
_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _-
_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `void std::gslice_array::operator/= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator/= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator/= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator/= (const _Tp &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator/= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `void std::gslice_array::operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator<<= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator<<= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator<<= (const valarray< _Tp > &)`

- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator<=<= (const _Expr< _Dom, _Tp`
`> &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator= (const _Expr< _Dom, _Tp > &)`
- `void std::slice_array::operator= (const valarray< _Tp > &) const`
- `void std::slice_array::operator= (const _Tp &) const`
- `template<class _Ex >`
`void std::mask_array::operator= (const _Expr< _Ex, _Tp > &__e) const`
- `void std::mask_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const valarray< _Tp > &)`
- `gslice_array & std::gslice_array::operator= (const gslice_array &)`
- `void std::gslice_array::operator= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator= (const slice_array< _Tp > &)`
- `void std::gslice_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const indirect_array< _Tp > &)`
- `gslice & std::gslice::operator= (const gslice &)`
- `indirect_array & std::indirect_array::operator= (const indirect_array &)`
- `void std::indirect_array::operator= (const valarray< _Tp > &) const`
- `slice_array & std::slice_array::operator= (const slice_array &)`
- `mask_array & std::mask_array::operator= (const mask_array &)`
- `void std::mask_array::operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `valarray & std::valarray::operator= (initializer_list< _Tp >)`
- `valarray< _Tp > & std::valarray::operator= (const gslice_array< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator= (const mask_array< _Tp > &)`

- `template<class _Dom >`
`void std::indirect_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator= (const _Tp &) const`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`

-
- ```

__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<
_Tp > &__v, const _Tp &__t)

```
- `template<typename _Tp >`  
`__Expr< __BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
  - `template<class _Dom >`  
`void std::gslice_array::operator>= (const _Expr< _Dom, _Tp > &) const`
  - `valarray< _Tp > & std::valarray::operator>= (const valarray< _Tp > &)`
  - `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator>= (const _Expr< _Dom, _Tp`  
`> &)`
  - `void std::gslice_array::operator>= (const valarray< _Tp > &) const`
  - `valarray< _Tp > & std::valarray::operator>= (const _Tp &)`
  - `indirect_array< _Tp > std::valarray::operator[] (const valarray< size_t > &)`
  - `slice_array< _Tp > std::valarray::operator[] (slice)`
  - `mask_array< _Tp > std::valarray::operator[] (const valarray< bool > &)`
  - `valarray< _Tp > std::valarray::operator[] (const valarray< bool > &) const`
  - `const _Tp & std::valarray::operator[] (size_t) const`
  - `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (const`  
`gslice &) const`
  - `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (slice)`  
`const`
  - `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (const`  
`valarray< size_t > &) const`
  - `_Tp & std::valarray::operator[] (size_t)`
  - `gslice_array< _Tp > std::valarray::operator[] (const gslice &)`
  - `template<typename _Tp >`  
`__Expr< __BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
  - `template<typename _Tp >`  
`__Expr< __BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
  - `template<typename _Tp >`  
`__Expr< __BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`  
`&__t, const valarray< _Tp > &__v)`
  - `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator^= (const _Expr< _Dom, _Tp >`  
`&)`
  - `valarray< _Tp > & std::valarray::operator^= (const _Tp &)`
-

- `template<class _Dom >`  
`void std::gslice_array::operator^ = (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator^ = (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator^ = (const valarray< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator| = (const _Expr< _Dom, _Tp >`  
`&)`
- `valarray< _Tp > & std::valarray::operator| = (const _Tp &)`
- `void std::gslice_array::operator| = (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator| = (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator| = (const valarray< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `_UnaryOp< __bitwise_not >::_Rt std::valarray::operator~ () const`
- `void std::valarray::resize (size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > std::valarray::shift (int) const`
- `size_t std::valarray::size () const`
- `size_t std::slice::size () const`
- `valarray< size_t > std::gslice::size () const`
- `size_t std::slice::start () const`
- `size_t std::gslice::start () const`

- `valarray< size_t > std::gslice::stride () const`
- `size_t std::slice::stride () const`
- `_Tp std::valarray::sum () const`

### 2.22.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

### 2.22.2 Function Documentation

#### 2.22.2.1 `std::gslice::gslice ( ) [inline, inherited]`

Construct an empty slice.

Definition at line 147 of file `gslice.h`.

#### 2.22.2.2 `std::gslice::gslice ( size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s ) [inline, inherited]`

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

#### Parameters

- o* Offset in array of first element.
- l* Array of dimension lengths.
- s* Array of dimension strides between array elements.

Definition at line 151 of file `gslice.h`.

#### 2.22.2.3 `std::gslice::gslice ( const gslice & __g ) [inline, inherited]`

Copy constructor.

Definition at line 156 of file `gslice.h`.

#### 2.22.2.4 `template<typename _Tp> std::gslice_array< _Tp >::gslice_array ( const gslice_array< _Tp > & __a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 142 of file `gslice_array.h`.

**2.22.2.5** `template<typename _Tp > std::indirect_array< _Tp  
>::indirect_array ( const indirect_array< _Tp > & __a )  
[inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 142 of file `indirect_array.h`.

**2.22.2.6** `template<typename _Tp > std::mask_array< _Tp >::mask_array ( const mask_array< _Tp > & a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 138 of file `mask_array.h`.

**2.22.2.7** `std::slice::slice ( ) [inline, inherited]`

Construct an empty slice.

Definition at line 89 of file `slice_array.h`.

**2.22.2.8** `std::slice::slice ( size_t __o, size_t __d, size_t __s ) [inline, inherited]`

Construct a slice.

#### Parameters

*o* Offset in array of first element.

*d* Number of elements in slice.

*s* Stride between array elements.

Definition at line 93 of file `slice_array.h`.

**2.22.2.9** `template<typename _Tp > std::slice_array< _Tp >::slice_array ( const slice_array< _Tp > & a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 206 of file `slice_array.h`.

**2.22.2.10** `template<typename _Tp> std::valarray< _Tp >::valarray ( const mask_array< _Tp > & __ma ) [inline, inherited]`

Construct an array with the same size and values in *ma*.

Definition at line 629 of file valarray.

**2.22.2.11** `template<typename _Tp> std::valarray<_Tp>::valarray ( const  
indirect_array<_Tp> & __ia ) [inline, inherited]`

Construct an array with the same size and values in *ia*.

Definition at line 638 of file valarray.

**2.22.2.12** `template<typename _Tp> std::valarray<_Tp>::valarray (   
initializer_list<_Tp> __l ) [inline, inherited]`

Construct an array with an [initializer\\_list](#) of values.

Definition at line 648 of file valarray.

**2.22.2.13** `template<typename _Tp> std::valarray<_Tp>::valarray ( )  
[inline, inherited]`

Construct an empty array.

Definition at line 577 of file valarray.

**2.22.2.14** `template<typename _Tp> std::valarray<_Tp>::valarray ( size_t  
__n ) [inline, explicit, inherited]`

Construct an array with *n* elements.

Definition at line 581 of file valarray.

**2.22.2.15** `template<typename _Tp> std::valarray<_Tp>::valarray ( const  
_Tp & __t, size_t __n ) [inline, inherited]`

Construct an array with *n* elements initialized to *t*.

Definition at line 587 of file valarray.

**2.22.2.16** `template<typename _Tp> std::valarray<_Tp>::valarray ( const  
valarray<_Tp> & __v ) [inline, inherited]`

Copy constructor.

Definition at line 602 of file valarray.

**2.22.2.17** `template<typename _Tp> std::valarray<_Tp>::valarray ( const slice_array<_Tp> & __sa ) [inline, inherited]`

Construct an array with the same size and values in *sa*.

Definition at line 609 of file valarray.

**2.22.2.18** `template<typename _Tp> std::valarray<_Tp>::valarray ( const gslice_array<_Tp> & __ga ) [inline, inherited]`

Construct an array with the same size and values in *ga*.

Definition at line 618 of file valarray.

**2.22.2.19** `std::gslice::~gslice ( ) [inline, inherited]`

Destructor.

Definition at line 161 of file gslice.h.

**2.22.2.20** `template<class _Tp> _Expr<_ValFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply ( _Tp func_Tp ) const [inline, inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying *func* to the corresponding element of this array. The new array has the same size as this one.

#### Parameters

*func* Function of *Tp* returning *Tp* to apply.

#### Returns

New valarray with transformed elements.

Definition at line 971 of file valarray.

**2.22.2.21** `template<class _Tp> _Expr<_RefFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply ( _Tp funcconst_Tp & ) const [inline, inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying *func* to the corresponding element of this array. The new array has the same size as this one.

**Parameters**

*func* Function of const Tp& returning Tp to apply.

**Returns**

New valarray with transformed elements.

Definition at line 979 of file valarray.

**2.22.2.22** `template<class _Tp > valarray<_Tp> std::valarray<_Tp>::cshift  
( int __n ) const [inline, inherited]`

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is  $(i - n) \% \text{size}()$ . The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

**Parameters**

*n* Number of element positions to rotate.

**Returns**

New valarray with elements in shifted positions.

Definition at line 897 of file valarray.

**2.22.2.23** `template<typename _Tp > _Tp std::valarray<_Tp>::max ( )  
const [inline, inherited]`

Return the maximum element using operator<().

Definition at line 963 of file valarray.

References std::max\_element().

**2.22.2.24** `template<typename _Tp > _Tp std::valarray<_Tp>::min ( )  
const [inline, inherited]`

Return the minimum element using operator<().

Definition at line 955 of file valarray.

References std::min\_element().

**2.22.2.25** `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__logical_not>::_Rt std::valarray<_Tp>::operator! ( ) const [inline, inherited]`

Return a new valarray by applying unary ! to each element.

Definition at line 998 of file valarray.

**2.22.2.26** `template<typename _Tp> void std::gslice_array<_Tp>::operator%=( const valarray<_Tp> & __v ) const [inline, inherited]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 201 of file gslice\_array.h.

**2.22.2.27** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator%=( const _Tp & __t ) [inline, inherited]`

Set each element *e* of array to *e % t*.

Definition at line 1025 of file valarray.

**2.22.2.28** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator%=( const valarray<_Tp> & __v ) [inline, inherited]`

Modulo elements of array by corresponding elements of *v*.

Definition at line 1025 of file valarray.

**2.22.2.29** `template<typename _Tp> void std::gslice_array<_Tp>::operator&=( const valarray<_Tp> & __v ) const [inline, inherited]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 205 of file gslice\_array.h.

**2.22.2.30** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator&=( const _Tp & __t ) [inline, inherited]`

Set each element *e* of array to *e & t*.

Definition at line 1027 of file valarray.



**2.22.2.31** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator&= ( const valarray<_Tp> & __v ) [inline, inherited]`

Logical and corresponding elements of *v* with elements of array.

Definition at line 1027 of file valarray.

**2.22.2.32** `template<typename _Tp> void std::gslice_array<_Tp>::operator*= ( const valarray<_Tp> & __v ) const [inline, inherited]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 199 of file gslice\_array.h.

**2.22.2.33** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator*= ( const _Tp & __t ) [inline, inherited]`

Multiply each element of array by *t*.

Definition at line 1023 of file valarray.

**2.22.2.34** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator*= ( const valarray<_Tp> & __v ) [inline, inherited]`

Multiply elements of array by corresponding elements of *v*.

Definition at line 1023 of file valarray.

**2.22.2.35** `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__unary_plus>::_Rt std::valarray<_Tp>::operator+ ( ) const [inline, inherited]`

Return a new valarray by applying unary + to each element.

Definition at line 995 of file valarray.

**2.22.2.36** `template<typename _Tp> void std::gslice_array<_Tp>::operator+= ( const valarray<_Tp> & __v ) const [inline, inherited]`

Add corresponding elements of *v* to slice elements.

Definition at line 202 of file gslice\_array.h.

**2.22.2.37** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator+=( const _Tp & __t ) [inline, inherited]`

Add  $t$  to each element of array.

Definition at line 1021 of file valarray.

**2.22.2.38** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator+=( const valarray<_Tp> & __v ) [inline, inherited]`

Add corresponding elements of  $v$  to elements of array.

Definition at line 1021 of file valarray.

**2.22.2.39** `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__negate>::_Rt std::valarray<_Tp>::operator-( ) const [inline, inherited]`

Return a new valarray by applying unary - to each element.

Definition at line 996 of file valarray.

**2.22.2.40** `template<typename _Tp> void std::gslice_array<_Tp>::operator-= ( const valarray<_Tp> & __v ) const [inline, inherited]`

Subtract corresponding elements of  $v$  from slice elements.

Definition at line 203 of file gslice\_array.h.

**2.22.2.41** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator-= ( const valarray<_Tp> & __v ) [inline, inherited]`

Subtract corresponding elements of  $v$  from elements of array.

Definition at line 1022 of file valarray.

**2.22.2.42** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator-= ( const _Tp & __t ) [inline, inherited]`

Subtract  $t$  to each element of array.

Definition at line 1022 of file valarray.

**2.22.2.43** `template<typename _Tp > void std::gslice_array< _Tp  
>::operator/= ( const valarray< _Tp > & __v ) const [inline,  
inherited]`

Divide slice elements by corresponding elements of  $v$ .

Definition at line 200 of file gslice\_array.h.

**2.22.2.44** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator/= ( const valarray< _Tp > & __v ) [inline,  
inherited]`

Divide elements of array by corresponding elements of  $v$ .

Definition at line 1024 of file valarray.

**2.22.2.45** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator/= ( const _Tp & __t ) [inline, inherited]`

Divide each element of array by  $t$ .

Definition at line 1024 of file valarray.

**2.22.2.46** `template<typename _Tp > void std::gslice_array< _Tp  
>::operator<<= ( const valarray< _Tp > & __v ) const  
[inline, inherited]`

Left shift slice elements by corresponding elements of  $v$ .

Definition at line 207 of file gslice\_array.h.

**2.22.2.47** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator<<= ( const _Tp & __t ) [inline, inherited]`

Left shift each element  $e$  of array by  $t$  bits.

Definition at line 1029 of file valarray.

**2.22.2.48** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator<<= ( const valarray< _Tp > & __v ) [inline, inherited]`

Left shift elements of array by corresponding elements of *v*.

Definition at line 1029 of file valarray.

**2.22.2.49** `template<typename _Tp > void std::slice_array< _Tp >::operator= ( const valarray< _Tp > & __v ) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 228 of file slice\_array.h.

**2.22.2.50** `template<typename _Tp > void std::slice_array< _Tp >::operator= ( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 223 of file slice\_array.h.

**2.22.2.51** `template<typename _Tp > void std::mask_array< _Tp >::operator= ( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 157 of file mask\_array.h.

**2.22.2.52** `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp >::operator= ( const valarray< _Tp > & __v ) [inline, inherited]`

Assign elements to an array.

Assign elements of array to values in *v*. Results are undefined if *v* does not have the same size as this array.

### Parameters

*v* Valarray to get values from.

Definition at line 669 of file valarray.

**2.22.2.53** `template<typename _Tp> gslice_array<_Tp> &  
std::gslice_array<_Tp>::operator=( const gslice_array<_Tp> &  
__a ) [inline, inherited]`

Assignment operator. Assigns slice elements to corresponding /// elements of *a*.

Definition at line 147 of file gslice\_array.h.

References `std::valarray<_Tp>::size()`.

**2.22.2.54** `template<typename _Tp> void std::gslice_array<_Tp>::operator=  
( const valarray<_Tp> & __v ) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 165 of file gslice\_array.h.

References `std::valarray<_Tp>::size()`.

**2.22.2.55** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>  
>::operator=( const _Tp & __t ) [inline, inherited]`

Assign elements to a value.

Assign all elements of array to *t*.

#### Parameters

*t* Value for elements.

Definition at line 717 of file valarray.

**2.22.2.56** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>  
>::operator=( const slice_array<_Tp> & __sa ) [inline,  
inherited]`

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

#### Parameters

*sa* Array slice to get values from.

Definition at line 725 of file valarray.

**2.22.2.57** `template<typename _Tp> void std::gslice_array<_Tp>::operator=( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 157 of file `gslice_array.h`.

References `std::valarray<_Tp>::size()`.

**2.22.2.58** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=( const indirect_array<_Tp> & __ia ) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

#### Parameters

*ia* Array slice to get values from.

Definition at line 755 of file `valarray`.

**2.22.2.59** `gslice & std::gslice::operator=( const gslice & __g ) [inline, inherited]`

Assignment operator.

Definition at line 168 of file `gslice.h`.

**2.22.2.60** `template<typename _Tp> indirect_array<_Tp> & std::indirect_array<_Tp>::operator=( const indirect_array<_Tp> & __a ) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 153 of file `indirect_array.h`.

**2.22.2.61** `template<typename _Tp> void std::indirect_array<_Tp>::operator=( const valarray<_Tp> & __v ) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 167 of file `indirect_array.h`.

**2.22.2.62** `template<typename _Tp> slice_array<_Tp> & std::slice_array<_Tp>::operator= ( const slice_array<_Tp> & __a ) [inline, inherited]`

Assignment operator. Assigns slice elements to corresponding /// elements of *a*.

Definition at line 214 of file slice\_array.h.

**2.22.2.63** `template<typename _Tp> mask_array<_Tp> & std::mask_array<_Tp>::operator= ( const mask_array<_Tp> & __a ) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 148 of file mask\_array.h.

**2.22.2.64** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator= ( initializer_list<_Tp> & __l ) [inline, inherited]`

Assign elements to an [initializer\\_list](#).

Assign elements of array to values in *l*. Results are undefined if *l* does not have the same size as this array.

#### Parameters

*l* [initializer\\_list](#) to get values from.

Definition at line 693 of file valarray.

**2.22.2.65** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator= ( const gsllice_array<_Tp> & __ga ) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

#### Parameters

*ga* Array slice to get values from.

Definition at line 735 of file valarray.

References `std::valarray<_Tp>::size()`.

**2.22.2.66** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=( const mask_array<_Tp> & __ma ) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

#### Parameters

*ma* Array slice to get values from.

Definition at line 745 of file valarray.

**2.22.2.67** `template<typename _Tp> void std::indirect_array<_Tp>::operator=( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 162 of file indirect\_array.h.

**2.22.2.68** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator>>=( const valarray<_Tp> & __v ) [inline, inherited]`

Right shift elements of array by corresponding elements of *v*.

Definition at line 1030 of file valarray.

**2.22.2.69** `template<typename _Tp> void std::gslice_array<_Tp>::operator>>=( const valarray<_Tp> & __v ) const [inline, inherited]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 208 of file gslice\_array.h.

**2.22.2.70** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator>>=( const _Tp & __t ) [inline, inherited]`

Right shift each element *e* of array by *t* bits.

Definition at line 1030 of file valarray.



**2.22.2.71** `template<typename _Tp > indirect_array< _Tp > std::valarray< _Tp >::operator[] ( const valarray< size_t > & __i ) [inline, inherited]`

Return a reference to an array subset.

Returns an [indirect\\_array](#) referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned [indirect\\_array](#) refers to these elements.

#### Parameters

*i* The valarray element index list.

#### Returns

Indirect\_array referencing elements in *i*.

Definition at line 835 of file valarray.

References std::valarray< \_Tp >::size().

**2.22.2.72** `template<typename _Tp > slice_array< _Tp > std::valarray< _Tp >::operator[] ( slice __s ) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

#### See also

[slice](#).

#### Parameters

*s* The source slice.

#### Returns

New valarray containing elements in *s*.

Definition at line 782 of file valarray.

**2.22.2.73** `template<typename _Tp > mask_array< _Tp > std::valarray< _Tp >::operator[] ( const valarray< bool > & __m ) [inline, inherited]`

Return a reference to an array subset.

Returns a new [mask\\_array](#) referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

### Parameters

*m* The valarray bitmask.

### Returns

New valarray containing elements indicated by *m*.

Definition at line 816 of file valarray.

References `std::valarray<_Tp>::size()`.

**2.22.2.74** `template<typename _Tp> valarray<_Tp> std::valarray<_Tp>::operator[] ( const valarray<bool> & __m ) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

### Parameters

*m* The valarray bitmask.

### Returns

New valarray containing elements indicated by *m*.

Definition at line 804 of file valarray.

References `std::valarray<_Tp>::size()`.

**2.22.2.75** `template<typename _Tp> _Expr<_GClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[] ( const gslice & __gs ) const [inline, inherited]`

Return an array subset.

Returns a [slice\\_array](#) referencing the elements of the array indicated by the slice argument.

**See also**

[gslice](#).

**Parameters**

*s* The source slice.

**Returns**

Slice\_array referencing elements indicated by *s*.

Definition at line 787 of file valarray.

**2.22.2.76** `template<typename _Tp > _Expr< _SClos< _ValArray, _Tp >, _Tp  
> std::valarray< _Tp >::operator[] ( slice __s ) const [inline,  
inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

[slice](#).

**Parameters**

*s* The source slice.

**Returns**

New valarray containing elements in *s*.

Definition at line 774 of file valarray.

**2.22.2.77** `template<typename _Tp > _Expr< _IClos< _ValArray, _Tp >, _Tp  
> std::valarray< _Tp >::operator[] ( const valarray< size_t > &  
__i ) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

**Parameters**

*i* The valarray element index list.

**Returns**

New valarray containing elements in *s*.

Definition at line 827 of file valarray.

**2.22.2.78** `template<typename _Tp > _Tp & std::valarray< _Tp >::operator[] ( size_t __i ) [inline, inherited]`

Return a reference to the *i*'th array element.

**Parameters**

*i* Index of element to return.

**Returns**

Reference to the *i*'th element.

Definition at line 551 of file valarray.

**2.22.2.79** `template<typename _Tp > gslice_array< _Tp > std::valarray< _Tp >::operator[] ( const gslice & __gs ) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the *gslice* argument. The new valarray has the same size as the input *gslice*.

**See also**

[gslice](#).

**Parameters**

*s* The source *gslice*.

**Returns**

New valarray containing elements in *s*.

Definition at line 796 of file valarray.

**2.22.2.80** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator^= ( const _Tp & __t ) [inline, inherited]`

Set each element *e* of array to  $e \wedge t$ .

Definition at line 1026 of file valarray.

**2.22.2.81** `template<typename _Tp > void std::gslice_array< _Tp  
>::operator^= ( const valarray< _Tp > & __v ) const [inline,  
inherited]`

Logical xor slice elements with corresponding elements of  $v$ .

Definition at line 204 of file `gslice_array.h`.

**2.22.2.82** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator^= ( const valarray< _Tp > & __v ) [inline,  
inherited]`

Logical xor corresponding elements of  $v$  with elements of array.

Definition at line 1026 of file `valarray`.

**2.22.2.83** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator|= ( const _Tp & __t ) [inline, inherited]`

Set each element  $e$  of array to  $e \mid t$ .

Definition at line 1028 of file `valarray`.

**2.22.2.84** `template<typename _Tp > void std::gslice_array< _Tp  
>::operator|= ( const valarray< _Tp > & __v ) const [inline,  
inherited]`

Logical or slice elements with corresponding elements of  $v$ .

Definition at line 206 of file `gslice_array.h`.

**2.22.2.85** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator|= ( const valarray< _Tp > & __v ) [inline,  
inherited]`

Logical or corresponding elements of  $v$  with elements of array.

Definition at line 1028 of file `valarray`.

**2.22.2.86** `template<typename _Tp > valarray< _Tp >::template _UnaryOp<  
__bitwise_not >::Rt std::valarray< _Tp >::operator~ ( ) const  
[inline, inherited]`

Return a new valarray by applying unary  $\sim$  to each element.

Definition at line 997 of file valarray.

**2.22.2.87** `template<class _Tp> void std::valarray<_Tp>::resize ( size_t  
__size, _Tp __c = _Tp() ) [inline, inherited]`

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

#### Parameters

- size* New array size.
- c* New value for all elements.

Definition at line 938 of file valarray.

**2.22.2.88** `template<class _Tp> valarray<_Tp> std::valarray<_Tp>::shift  
( int __n ) const [inline, inherited]`

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

#### Parameters

- n* Number of element positions to shift.

#### Returns

- New valarray with elements in shifted positions.

Definition at line 856 of file valarray.

**2.22.2.89** `template<class _Tp> size_t std::valarray<_Tp>::size ( ) const  
[inline, inherited]`

Return the number of elements in array.

Definition at line 843 of file valarray.

Referenced by `std::valarray<_Tp>::operator=()`, `std::gslice_array<_Tp>::operator=()`, and `std::valarray<_Tp>::operator[]()`.

**2.22.2.90** `size_t std::slice::size ( ) const [inline, inherited]`

Return size of slice.

Definition at line 101 of file slice\_array.h.

**2.22.2.91** `valarray< size_t > std::gslice::size ( ) const [inline, inherited]`

Return array of sizes of slice dimensions.

Definition at line 137 of file gslice.h.

**2.22.2.92** `size_t std::slice::start ( ) const [inline, inherited]`

Return array offset of first slice element.

Definition at line 97 of file slice\_array.h.

**2.22.2.93** `size_t std::gslice::start ( ) const [inline, inherited]`

Return array offset of first slice element.

Definition at line 133 of file gslice.h.

**2.22.2.94** `valarray< size_t > std::gslice::stride ( ) const [inline, inherited]`

Return array of array strides for each dimension.

Definition at line 141 of file gslice.h.

**2.22.2.95** `size_t std::slice::stride ( ) const [inline, inherited]`

Return array stride of slice.

Definition at line 105 of file slice\_array.h.

**2.22.2.96** `template<class _Tp > _Tp std::valarray< _Tp >::sum ( ) const [inline, inherited]`

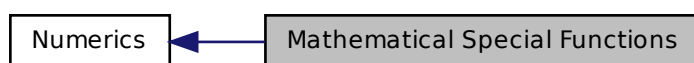
Return the sum of all elements in the array.

Accumulates the sum of all elements into a `Tp` using `+=`. The order of adding the elements is unspecified.

Definition at line 848 of file valarray.

## 2.23 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float std::tr1::betaf` (float \_\_x, float \_\_y)
- `long double std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_1f` (float \_\_k)
- `long double std::tr1::comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_2f` (float \_\_k)



- long double **std::tr1::comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type **std::tr1::comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **std::tr1::comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **std::tr1::comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type **std::tr1::conf\_hyperg** (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::conf\_hypergf** (float \_\_a, float \_\_c, float \_\_x)
- long double **std::tr1::conf\_hypergl** (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_i** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_j** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_k** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)

- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint\_3 (_Tp`  
`__k, _Tpn __nu, _Tpp __phi)`
- `float std::tr1::ellint\_3f (float __k, float __nu, float __phi)`
- `long double std::tr1::ellint\_3l (long double __k, long double __nu, long double`  
`__phi)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)`
- `float std::tr1::expintf (float __x)`
- `long double std::tr1::expintl (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp`  
`__x)`
- `float std::tr1::hermitef (unsigned int __n, float __x)`
- `long double std::tr1::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg`  
`( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`
- `long double std::tr1::hypergl (long double __a, long double __b, long double`  
`__c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _-`  
`Tp __x)`
- `float std::tr1::laguerref (unsigned int __n, float __x)`
- `long double std::tr1::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _-`  
`Tp __x)`
- `float std::tr1::legendref (unsigned int __n, float __x)`
- `long double std::tr1::legendrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann\_zeta (_Tp __x)`
- `float std::tr1::riemann\_zetaf (float __x)`
- `long double std::tr1::riemann\_zetal (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_bessel (unsigned int __n,`  
`_Tp __x)`
- `float std::tr1::sph\_besself (unsigned int __n, float __x)`
- `long double std::tr1::sph\_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_legendre (unsigned int _-`  
`_l, unsigned int __m, _Tp __theta)`

- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp > \_\_gnu\_cxx::\_\_promote<\_Tp>::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

### 2.23.1 Detailed Description

A collection of advanced mathematical special functions.

### 2.23.2 Function Documentation

**2.23.2.1** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::assoc_laguerre ( unsigned int __n, unsigned int __m, _Tp  
__x ) [inline]`

5.2.1.1 Associated Laguerre polynomials.

Definition at line 123 of file tr1/cmath.

**2.23.2.2** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::assoc_legendre ( unsigned int __l, unsigned int __m, _Tp  
__x ) [inline]`

5.2.1.2 Associated Legendre functions.

Definition at line 140 of file tr1/cmath.

**2.23.2.3** `template<typename _Tpx , typename _Tpy >  
__gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta ( _Tpx  
__x, _Tpy __y ) [inline]`

5.2.1.3 Beta functions.

Definition at line 157 of file tr1/cmath.

**2.23.2.4** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::comp_ellint_1( _Tp __k ) [inline]`

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 174 of file tr1/cmath.

**2.23.2.5** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::comp_ellint_2( _Tp __k ) [inline]`

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 191 of file tr1/cmath.

**2.23.2.6** `template<typename _Tp , typename _Tpn > __gnu_cxx::__promote_  
2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3( _Tp __k, _Tpn  
__nu ) [inline]`

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 208 of file tr1/cmath.

**2.23.2.7** `template<typename _Tpa , typename _Tpc , typename _Tp  
> __gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type  
std::tr1::conf_hyperg( _Tpa __a, _Tpc __c, _Tp __x ) [inline]`

5.2.1.7 Confluent hypergeometric functions.

Definition at line 225 of file tr1/cmath.

**2.23.2.8** `template<typename _Tpnu , typename _Tp >  
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i  
( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 242 of file tr1/cmath.

**2.23.2.9** `template<typename _Tpnu , typename _Tp >  
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j  
( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 259 of file tr1/cmath.

**2.23.2.10** `template<typename _Tpnu , typename _Tp >  
 __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type  
 std::tr1::cyl_bessel_k ( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 276 of file tr1/cmath.

**2.23.2.11** `template<typename _Tpnu , typename _Tp >  
 __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type  
 std::tr1::cyl_neumann ( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.11 Cylindrical Neumann functions.

Definition at line 293 of file tr1/cmath.

**2.23.2.12** `template<typename _Tp , typename _Tpp >  
 __gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (   
 _Tp __k, _Tpp __phi ) [inline]`

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 310 of file tr1/cmath.

**2.23.2.13** `template<typename _Tp , typename _Tpp >  
 __gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (   
 _Tp __k, _Tpp __phi ) [inline]`

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 327 of file tr1/cmath.

**2.23.2.14** `template<typename _Tp , typename _Tpn , typename _Tpp  
 > __gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type  
 std::tr1::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi ) [inline]`

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 344 of file tr1/cmath.

**2.23.2.15** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
 std::tr1::expint ( _Tp __x ) [inline]`

5.2.1.15 Exponential integrals.

Definition at line 361 of file tr1/cmath.

**2.23.2.16** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::hermite ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.16 Hermite polynomials.

Definition at line 378 of file tr1/cmath.

**2.23.2.17** `template<typename _Tpa , typename _Tpb , typename _Tpc ,  
typename _Tp > __gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc,  
_Tp>::__type std::tr1::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c,  
_Tp __x ) [inline]`

5.2.1.17 Hypergeometric functions.

Definition at line 395 of file tr1/cmath.

**2.23.2.18** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::laguerre ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.18 Laguerre polynomials.

Definition at line 412 of file tr1/cmath.

**2.23.2.19** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::legendre ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.19 Legendre polynomials.

Definition at line 429 of file tr1/cmath.

**2.23.2.20** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::riemann_zeta ( _Tp __x ) [inline]`

5.2.1.20 Riemann zeta function.

Definition at line 446 of file tr1/cmath.

**2.23.2.21** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::sph_bessel ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.21 Spherical Bessel functions.

Definition at line 463 of file tr1/cmath.

**2.23.2.22** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::sph_legendre ( unsigned int __l, unsigned int __m, _Tp  
__theta ) [inline]`

5.2.1.22 Spherical associated Legendre functions.

Definition at line 480 of file tr1/cmath.

**2.23.2.23** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::sph_neumann ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.23 Spherical Neumann functions.

Definition at line 497 of file tr1/cmath.

## 2.24 Type Traits

Collaboration diagram for Type Traits:



### Classes

- struct `std::__declval_protector<_Tp >`  
*declval*
- struct `std::add_lvalue_reference<_Tp >`  
*add\_lvalue\_reference*
- struct `std::add_rvalue_reference<_Tp >`  
*add\_rvalue\_reference*
- struct `std::aligned_storage<_Len, _Align >`  
*Alignment type.*

- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`  
*conditional*
- class `std::decay< _Tp >`  
*decay*
- struct `std::enable_if< bool, _Tp >`  
*enable\_if*
- struct `std::has_nothrow_copy_assign< _Tp >`  
*has\_nothrow\_copy\_assign*
- struct `std::has_nothrow_copy_constructor< _Tp >`  
*has\_nothrow\_copy\_constructor*
- struct `std::has_nothrow_default_constructor< _Tp >`  
*has\_nothrow\_default\_constructor*
- struct `std::has_trivial_copy_assign< _Tp >`  
*has\_trivial\_copy\_assign*
- struct `std::has_trivial_copy_constructor< _Tp >`  
*has\_trivial\_copy\_constructor*
- struct `std::has_trivial_default_constructor< _Tp >`  
*has\_trivial\_default\_constructor*
- struct `std::has_trivial_destructor< _Tp >`  
*has\_trivial\_destructor*
- struct `std::is_base_of< _Base, _Derived >`  
*is\_base\_of*
- struct `std::is_constructible< _Tp, _Args >`  
*is\_constructible*
- struct `std::is_convertible< _From, _To >`  
*is\_convertible*
- struct `std::is_explicitly_convertible< _From, _To >`  
*is\_explicitly\_convertible*



- struct `std::is_lvalue_reference< typename >`  
*is\_lvalue\_reference*
- struct `std::is_nothrow_constructible< _Tp, _Args >`  
*is\_nothrow\_constructible*
- struct `std::is_pod< _Tp >`  
*is\_pod*
- struct `std::is_reference< _Tp >`  
*is\_reference*
- struct `std::is_rvalue_reference< typename >`  
*is\_rvalue\_reference*
- struct `std::is_signed< _Tp >`  
*is\_signed*
- struct `std::is_standard_layout< _Tp >`  
*is\_standard\_layout*
- struct `std::is_trivial< _Tp >`  
*is\_trivial*
- struct `std::is_unsigned< _Tp >`  
*is\_unsigned*
- struct `std::make_signed< _Tp >`  
*make\_signed*
- struct `std::make_unsigned< _Tp >`  
*make\_unsigned*
- struct `std::remove_reference< _Tp >`  
*remove\_reference*
- struct `std::tr1::__is_member_pointer_helper< _Tp >`  
*is\_member\_pointer*
- struct `std::tr1::add_const< _Tp >`  
*add\_const*

- struct `std::tr1::add_cv< _Tp >`  
*add\_cv*
- struct `std::tr1::add_pointer< _Tp >`  
*add\_pointer*
- struct `std::tr1::add_volatile< _Tp >`  
*add\_volatile*
- struct `std::tr1::alignment_of< _Tp >`  
*alignment\_of*
- struct `std::tr1::extent< typename, _Uint >`  
*extent*
- struct `std::tr1::has_virtual_destructor< _Tp >`  
*has\_virtual\_destructor*
- struct `std::tr1::integral_constant< _Tp, __v >`  
*integral\_constant*
- struct `std::tr1::is_abstract< _Tp >`  
*is\_abstract*
- struct `std::tr1::is_arithmetic< _Tp >`  
*is\_arithmetic*
- struct `std::tr1::is_array< typename >`  
*is\_array*
- struct `std::tr1::is_class< _Tp >`  
*is\_class*
- struct `std::tr1::is_compound< _Tp >`  
*is\_compound*
- struct `std::tr1::is_const< typename >`  
*is\_const*
- struct `std::tr1::is_empty< _Tp >`  
*is\_empty*

- struct `std::tr1::is_enum< _Tp >`  
*is\_enum*
- struct `std::tr1::is_floating_point< _Tp >`  
*is\_floating\_point*
- struct `std::tr1::is_function< typename >`  
*is\_function*
- struct `std::tr1::is_fundamental< _Tp >`  
*is\_fundamental*
- struct `std::tr1::is_integral< _Tp >`  
*is\_integral*
- struct `std::tr1::is_member_function_pointer< _Tp >`  
*is\_member\_function\_pointer*
- struct `std::tr1::is_member_object_pointer< _Tp >`  
*is\_member\_object\_pointer*
- struct `std::tr1::is_object< _Tp >`  
*is\_object*
- struct `std::tr1::is_pointer< _Tp >`  
*is\_pointer*
- struct `std::tr1::is_polymorphic< _Tp >`  
*is\_polymorphic*
- struct `std::tr1::is_same< typename, typename >`  
*is\_same*
- struct `std::tr1::is_scalar< _Tp >`  
*is\_scalar*
- struct `std::tr1::is_union< _Tp >`  
*is\_union*
- struct `std::tr1::is_void< _Tp >`  
*is\_void*

- struct `std::tr1::is_volatile< typename >`  
*is\_volatile*
- struct `std::tr1::rank< typename >`  
*rank*
- struct `std::tr1::remove_all_extents< _Tp >`  
*remove\_all\_extents*
- struct `std::tr1::remove_const< _Tp >`  
*remove\_const*
- struct `std::tr1::remove_cv< _Tp >`  
*remove\_cv*
- struct `std::tr1::remove_extent< _Tp >`  
*remove\_extent*
- struct `std::tr1::remove_pointer< _Tp >`  
*remove\_pointer*
- struct `std::tr1::remove_volatile< _Tp >`  
*remove\_volatile*

## Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`

## Typedefs

- `typedef integral_constant< bool, false > std::tr1::false_type`
- `typedef integral_constant< bool, true > std::tr1::true_type`

## Functions

- `template<typename _Tp >`  
`add_rvalue_reference< _Tp >::type std::declval () noexcept`

## Variables

- static const `_Tp std::tr1::integral_constant::value`

### 2.24.1 Detailed Description

Compile time type transformation and information.

### 2.24.2 Typedef Documentation

#### 2.24.2.1 `typedef integral_constant<bool, false> std::tr1::false_type`

typedef for `false_type`

Definition at line 78 of file `tr1_impl/type_traits`.

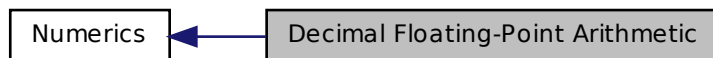
#### 2.24.2.2 `typedef integral_constant<bool, true> std::tr1::true_type`

typedef for `true_type`

Definition at line 75 of file `tr1_impl/type_traits`.

## 2.25 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



## Namespaces

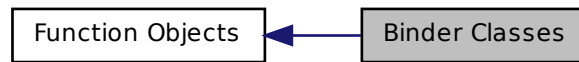
- namespace [`std::decimal`](#)

### 2.25.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

## 2.26 Binder Classes

Collaboration diagram for Binder Classes:



### Classes

- class `std::binder1st< _Operation >`  
*One of the [binder functors](#).*
- class `std::binder2nd< _Operation >`  
*One of the [binder functors](#).*
- struct `std::is_bind_expression< _Tp >`  
*Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].*
- struct `std::is_bind_expression< _Bind< _Signature > >`  
*Class template `_Bind` is always a bind expression.*
- struct `std::is_bind_expression< _Bind_result< _Result, _Signature > >`  
*Class template `_Bind` is always a bind expression.*
- struct `std::is_placeholder< _Tp >`  
*Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct `std::is_placeholder< _Placeholder< _Num > >`

## Namespaces

- namespace `std::placeholders`

## Functions

- `template<typename _Functor , typename... _ArgTypes>`  
`_Bind< typename _Maybe_wrap_member_pointer< _Functor >::type(_-`  
`ArgTypes...)> std::bind (_Functor __f, _ArgTypes...__args)`
- `template<typename _Operation , typename _Tp >`  
`binder1st< _Operation > std::binder1st (const _Operation &__fn, const _Tp &_-`  
`__x)`
- `template<typename _Operation , typename _Tp >`  
`binder2nd< _Operation > std::binder2nd (const _Operation &__fn, const _Tp &_-`  
`__x)`

## Variables

- `std::binder1st` `std::_GLIBCXX_DEPRECATED_ATTR`

### 2.26.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B's operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be called with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `binder1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>, 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `binder1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `binder1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `binder1st(std::plus<int>, 5)`.

These become more useful when combined with the composition functions.

## 2.26.2 Function Documentation

**2.26.2.1** `template<typename _Functor , typename... _ArgTypes>  
_Bind<typename _Maybe_wrap_member_pointer<_  
_Functor>::type(_ArgTypes...)> std::bind ( _Functor __f,  
_ArgTypes... __args ) [inline]`

Function template for `std::bind`.

Definition at line 1351 of file `functional`.

**2.26.2.2** `template<typename _Operation , typename _Tp >  
binder1st<_Operation> std::bind1st ( const _Operation & __fn,  
const _Tp & __x ) [inline]`

One of the [binder functions](#).

Definition at line 125 of file `binders.h`.

**2.26.2.3** `template<typename _Operation , typename _Tp >  
binder2nd<_Operation> std::bind2nd ( const _Operation & __fn,  
const _Tp & __x ) [inline]`

One of the [binder functions](#).

Definition at line 160 of file `binders.h`.

## 2.26.3 Variable Documentation

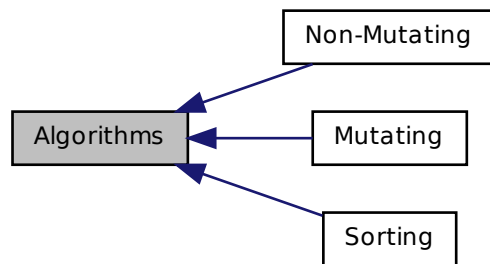
**2.26.3.1** `std::binder2nd std::_GLIBCXX_DEPRECATED_ATTR`

One of the [binder functions](#).



## 2.27 Algorithms

Collaboration diagram for Algorithms:



### Modules

- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

#### 2.27.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

## 2.28 Mutating

Collaboration diagram for Mutating:



### Functions

- `template<typename _II, typename _OI >`  
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator`  
`__result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`  
`value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`  
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator`  
`__gen)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`  
`__pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`

- `template<typename _Tp >`  
`std::remove_reference< _Tp >::type && std::move ( _Tp &&__t)`
- `template<typename _BI1 , typename _BI2 >`  
`_BI2 std::move_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::partition ( _ForwardIterator __first, _ForwardIterator __-`  
`last, _Predicate __pred)`
- `template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , type-`  
`name _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy ( _InputIterator`  
`__first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`  
`out_false, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::partition_point ( _ForwardIterator __first, _ForwardIterator`  
`__last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::random_shuffle ( _RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _RandomNumberGenerator >`  
`void std::random_shuffle ( _RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator , typename _Tp >`  
`_ForwardIterator std::remove ( _ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Tp >`  
`_OutputIterator std::remove_copy ( _InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate >`  
`_OutputIterator std::remove_copy_if ( _InputIterator __first, _InputIterator __-`  
`last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::remove_if ( _ForwardIterator __first, _ForwardIterator __-`  
`last, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Tp >`  
`void std::replace ( _ForwardIterator __first, _ForwardIterator __last, const _Tp`  
`&__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _-`  
`Tp >`  
`_OutputIterator std::replace_copy_if ( _InputIterator __first, _InputIterator __-`  
`last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator , typename _Predicate , typename _Tp >`  
`void std::replace_if ( _ForwardIterator __first, _ForwardIterator __last, _-`  
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`void std::reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`  
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __-`  
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _UniformRandomNumberGenerator &__g)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _-`  
`ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_`  
`op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`  
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred)`

## 2.28.1 Function Documentation

**2.28.1.1** `template<typename _II, typename _OI> _OI std::copy ( _II __first,  
_II __last, _OI __result ) [inline]`

Copies the range [first,last) into result.

### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

### Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 463 of file `stl_algobase.h`.

**2.28.1.2** `template<typename _BI1, typename _BI2> _BI2  
std::copy_backward ( _BI1 __first, _BI1 __last, _BI2 __result )  
[inline]`

Copies the range [first,last) into result.

### Parameters

*first* A bidirectional iterator.

*last* A bidirectional iterator.

*result* A bidirectional iterator.

### Returns

result - (first - last)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are

passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use copy instead. Note that the start of the output range may overlap [first,last).

Definition at line 632 of file stl\_algobase.h.

**2.28.1.3** `template<typename _InputIterator , typename _OutputIterator ,  
typename _Predicate > _OutputIterator std::copy_if ( _InputIterator  
__first, _InputIterator __last, _OutputIterator __result, _Predicate  
__pred )`

Copy the elements of a sequence for which a predicate is true.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*pred* A predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) for which *pred* returns true to the range beginning at *result*.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 949 of file stl\_algo.h.

**2.28.1.4** `template<typename _InputIterator , typename _Size , typename  
_OutputIterator > _OutputIterator std::copy_n ( _InputIterator  
__first, _Size __n, _OutputIterator __result ) [inline]`

Copies the range [first,first+n) into [result,result+n).

#### Parameters

*first* An input iterator.

*n* The number of elements to copy.

*result* An output iterator.

**Returns**

result+n.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1006 of file `stl_algo.h`.

References `std::__iterator_category()`.

**2.28.1.5** `template<typename _ForwardIterator, typename _Tp> void std::fill  
( _ForwardIterator __first, _ForwardIterator __last, const _Tp &  
__value ) [inline]`

Fills the range `[first,last)` with copies of `value`.

**Parameters**

*first* A forward iterator.

*last* A forward iterator.

*value* A reference-to-const of arbitrary type.

**Returns**

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 734 of file `stl_algobase.h`.

**2.28.1.6** `template<typename _OI, typename _Size, typename _Tp> _OI  
std::fill_n ( _OI __first, _Size __n, const _Tp & __value )  
[inline]`

Fills the range `[first,first+n)` with copies of `value`.

**Parameters**

*first* An output iterator.

*n* The count of copies to perform.

*value* A reference-to-const of arbitrary type.

**Returns**

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 794 of file `stl_algobase.h`.

**2.28.1.7** `template<typename _ForwardIterator, typename _Generator> void  
std::generate ( _ForwardIterator __first, _ForwardIterator __last,  
_Generator __gen )`

Assign the result of a function object to each value in a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*gen* A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

#### Returns

`generate()` returns no value.

Performs the assignment `*i = gen ()` for each `i` in the range `[first,last)`.

Definition at line 4809 of file `stl_algo.h`.

**2.28.1.8** `template<typename _OutputIterator, typename _Size, typename  
_Generator> _OutputIterator std::generate_n ( _OutputIterator  
__first, _Size __n, _Generator __gen )`

Assign the result of a function object to each value in a sequence.

#### Parameters

*first* A forward iterator.

*n* The length of the sequence.

*gen* A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

#### Returns

The end of the sequence, `first+n`



Performs the assignment `*i = gen()` for each `i` in the range `[first, first+n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 4840 of file `stl_algo.h`.

**2.28.1.9** `template<typename _InputIterator, typename _Predicate> bool  
std::is_partitioned ( _InputIterator __first, _InputIterator __last,  
_Predicate __pred ) [inline]`

Checks whether the sequence is partitioned.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

True if the range `[first, last)` is partitioned by `pred`, i.e. if all elements that satisfy `pred` appear before those that do not.

Definition at line 801 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

**2.28.1.10** `template<typename _ForwardIterator1, typename  
_ForwardIterator2> void std::iter_swap ( _ForwardIterator1 __a,  
_ForwardIterator2 __b ) [inline]`

Swaps the contents of two iterators.

#### Parameters

*a* An iterator.

*b* Another iterator.

#### Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 116 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::next_permutation()`, `std::prev_permutation()`, `std::random_shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

**2.28.1.11** `template<typename _II, typename _OI> _OI std::move ( _II  
__first, _II __last, _OI __result ) [inline]`

Moves the range `[first,last)` into `result`.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

#### Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 496 of file `stl_algobase.h`.

**2.28.1.12** `template<typename _Tp> std::remove_reference<_Tp>::type&&  
std::move ( _Tp && __t ) [inline]`

Move a value.

#### Parameters

*\_\_t* A thing of arbitrary type.

#### Returns

Same, moved.

Definition at line 93 of file `move.h`.

**2.28.1.13** `template<typename _BI1 , typename _BI2 > _BI2  
std::move_backward ( _BI1 __first, _BI1 __last, _BI2 __result )  
[inline]`

Moves the range [first,last) into result.

#### Parameters

*first* A bidirectional iterator.

*last* A bidirectional iterator.

*result* A bidirectional iterator.

#### Returns

result - (first - last)

The function has the same effect as move, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to memmove whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use move instead. Note that the start of the output range may overlap [first,last).

Definition at line 668 of file stl\_algobase.h.

**2.28.1.14** `template<typename _ForwardIterator , typename _Predicate  
> _ForwardIterator std::partition ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred ) [inline]`

Move elements for which a predicate is true to the beginning of a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate functor.

#### Returns

An iterator `middle` such that `pred(i)` is true for each iterator `i` in the range [first,middle) and false for each `i` in the range [middle,last).

`pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 5012 of file stl\_algo.h.

References `std::__iterator_category()`, and `std::__partition()`.

**2.28.1.15** `template<typename _InputIterator , typename _OutputIterator1  
, typename _OutputIterator2 , typename _Predicate >  
pair<_OutputIterator1, _OutputIterator2> std::partition_copy (`  
`_inputIterator __first, _InputIterator __last, _OutputIterator1`  
`__out_true, _OutputIterator2 __out_false, _Predicate __pred )`

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*out\_true* An output iterator.  
*out\_false* An output iterator.  
*pred* A predicate.

#### Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range [first,last) for which `pred` returns true to the range beginning at `out_true` and each element for which `pred` returns false to `out_false`.

Definition at line 1035 of file `stl_algo.h`.

**2.28.1.16** `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::partition_point ( _ForwardIterator __first,`  
`_ForwardIterator __last, _Predicate __pred )`

Find the partition point of a partitioned range.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*pred* A predicate.

#### Returns

An iterator `mid` such that `all_of(first, mid, pred)` and `none_of(mid, last, pred)` are both true.

Definition at line 819 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

**2.28.1.17** `template<typename _RandomAccessIterator > void  
std::random_shuffle ( _RandomAccessIterator __first,  
_RandomAccessIterator __last ) [inline]`

Randomly shuffle the elements of a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

#### Returns

Nothing.

Reorder the elements in the range [first,last) using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 4948 of file stl\_algo.h.

References std::iter\_swap().

**2.28.1.18** `template<typename _RandomAccessIterator , typename  
_RandomNumberGenerator > void std::random_shuffle (  
_RandomAccessIterator __first, _RandomAccessIterator __last,  
_RandomNumberGenerator && __rand )`

Shuffle the elements of a sequence using a random number generator.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*rand* The RNG functor or function.

#### Returns

Nothing.

Reorders the elements in the range [first,last) using *rand* to provide a random distribution. Calling *rand*(N) for a positive integer N should return a randomly chosen integer from the range [0,N).

Definition at line 4976 of file stl\_algo.h.

References std::iter\_swap().

**2.28.1.19** `template<typename _ForwardIterator, typename _Tp >  
 _ForwardIterator std::remove ( _ForwardIterator __first,  
 _ForwardIterator __last, const _Tp & __value )`

Remove elements from a sequence.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*value* The value to be removed.

#### Returns

An iterator designating the end of the resulting sequence.

All elements equal to *value* are removed from the range [first,last).

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and *last* are still present, but their value is unspecified.

Definition at line 1084 of file `stl_algo.h`.

**2.28.1.20** `template<typename _InputIterator, typename _OutputIterator,  
 typename _Tp > _OutputIterator std::remove_copy ( _InputIterator  
 __first, _InputIterator __last, _OutputIterator __result, const _Tp  
 & __value )`

Copy a sequence, removing elements of a given value.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*result* An output iterator.  
*value* The value to be removed.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) not equal to *value* to the range beginning at *result*. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 872 of file `stl_algo.h`.

**2.28.1.21** `template<typename _InputIterator , typename _OutputIterator  
 , typename _Predicate > _OutputIterator std::remove_copy_if (  
 _InputIterator __first, _InputIterator __last, _OutputIterator  
 __result, _Predicate __pred )`

Copy a sequence, removing elements for which a predicate is true.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*pred* A predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) for which `pred` returns false to the range beginning at `result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 910 of file `stl_algo.h`.

**2.28.1.22** `template<typename _ForwardIterator , typename _Predicate >  
 _ForwardIterator std::remove_if ( _ForwardIterator __first,  
 _ForwardIterator __last, _Predicate __pred )`

Remove elements from a sequence using a predicate.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate.

#### Returns

An iterator designating the end of the resulting sequence.

All elements for which `pred` returns true are removed from the range [first,last).

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1127 of file `stl_algo.h`.

**2.28.1.23** `template<typename _ForwardIterator, typename _Tp> void  
std::replace ( _ForwardIterator __first, _ForwardIterator __last,  
const _Tp & __old_value, const _Tp & __new_value )`

Replace each occurrence of one value in a sequence with another value.

#### Parameters

*first* A forward iterator.  
*last* A forward iterator.  
*old\_value* The value to be replaced.  
*new\_value* The replacement value.

#### Returns

`replace()` returns no value.

For each iterator `i` in the range `[first,last)` if `*i == old_value` then the assignment `*i = new_value` is performed.

Definition at line 4745 of file `stl_algo.h`.

**2.28.1.24** `template<typename _InputIterator, typename _OutputIterator  
, typename _Predicate, typename _Tp> _OutputIterator  
std::replace_copy_if ( _InputIterator __first, _InputIterator  
__last, _OutputIterator __result, _Predicate __pred, const _Tp &  
__new_value )`

Copy a sequence, replacing each value for which a predicate returns true with another value.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*result* An output iterator.  
*pred* A predicate.  
*new\_value* The replacement value.



**Returns**

The end of the output sequence, `result+(last-first)`.

Copies each element in the range `[first,last)` to the range `[result,result+(last-first))` replacing elements for which `pred` returns true with `new_value`.

Definition at line 3786 of file `stl_algo.h`.

**2.28.1.25** `template<typename _ForwardIterator , typename _Predicate ,  
typename _Tp > void std::replace_if ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred, const _Tp &  
__new_value )`

Replace each value in a sequence for which a predicate returns true with another value.

**Parameters**

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate.

*new\_value* The replacement value.

**Returns**

`replace_if()` returns no value.

For each iterator `i` in the range `[first,last)` if `pred(*i)` is true then the assignment `*i = new_value` is performed.

Definition at line 4777 of file `stl_algo.h`.

**2.28.1.26** `template<typename _BidirectionalIterator > void std::reverse (`  
`_BidirectionalIterator __first, _BidirectionalIterator __last )`  
`[inline]`

Reverse a sequence.

**Parameters**

*first* A bidirectional iterator.

*last* A bidirectional iterator.

**Returns**

`reverse()` returns no value.

Reverses the order of the elements in the range  $[first, last)$ , so that the first element becomes the last etc. For every  $i$  such that  $0 \leq i \leq (last - first) / 2$ , `reverse()` swaps  $*(first + i)$  and  $*(last - (i + 1))$ .

Definition at line 1435 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

**2.28.1.27** `template<typename _BidirectionalIterator, typename  
_OutputIterator> _OutputIterator std::reverse_copy (  
_BidirectionalIterator __first, _BidirectionalIterator __last,  
_OutputIterator __result )`

Copy a sequence, reversing its elements.

#### Parameters

*first* A bidirectional iterator.

*last* A bidirectional iterator.

*result* An output iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range  $[first, last)$  to the range  $[result, result + (last - first))$  such that the order of the elements is reversed. For every  $i$  such that  $0 \leq i \leq (last - first)$ , `reverse_copy()` performs the assignment  $*(result + (last - first) - i) = *(first + i)$ . The ranges  $[first, last)$  and  $[result, result + (last - first))$  must not overlap.

Definition at line 1462 of file `stl_algo.h`.

**2.28.1.28** `template<typename _ForwardIterator> void std::rotate  
( _ForwardIterator __first, _ForwardIterator __middle,  
_ForwardIterator __last ) [inline]`

Rotate the elements of a sequence.

#### Parameters

*first* A forward iterator.

*middle* A forward iterator.

*last* A forward iterator.

**Returns**

Nothing.

Rotates the elements of the range `[first,last)` by `(middle-first)` positions so that the element at `middle` is moved to `first`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

This effectively swaps the ranges `[first,middle)` and `[middle,last)`.

Performs `*(first+(n+(last-middle))%(last-first))=*(first+n)` for each `n` in the range `[0,last-first)`.

Definition at line 1666 of file `stl_algo.h`.

References `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__stable_partition_adaptive()`.

```
2.28.1.29 template<typename _ForwardIterator , typename _OutputIterator
 > _OutputIterator std::rotate_copy (_ForwardIterator
 __first, _ForwardIterator __middle, _ForwardIterator __last,
 _OutputIterator __result)
```

Copy a sequence, rotating its elements.

**Parameters**

*first* A forward iterator.

*middle* A forward iterator.

*last* A forward iterator.

*result* An output iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[first,last)` to the range beginning at

**Returns**

, rotating the copied elements by `(middle-first)` positions so that the element at `middle` is moved to `result`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

Performs `*(result+(n+(last-middle))%(last-first))=*(first+n)` for each `n` in the range `[0,last-first)`.

Definition at line 1700 of file `stl_algo.h`.

**2.28.1.30** `template<typename _RandomAccessIterator , typename  
_UniformRandomNumberGenerator > void std::shuffle (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_UniformRandomNumberGenerator & __g )`

Shuffle the elements of a sequence using a uniform random number generator.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*g* A UniformRandomNumberGenerator (26.5.1.3).

#### Returns

Nothing.

Reorders the elements in the range [first,last) using *g* to provide random numbers.

Definition at line 4135 of file `stl_algo.h`.

References `std::iter_swap()`.

**2.28.1.31** `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::stable_partition ( _ForwardIterator __first,`  
`_ForwardIterator __last, _Predicate __pred )`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate functor.

#### Returns

An iterator `middle` such that `pred(i)` is true for each iterator *i* in the range [first,middle) and false for each *i* in the range [middle,last).

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements *x* and *y* in the range [first,last) such that `pred(x)==pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1858 of file `stl_algo.h`.

References `std::__inplace_stable_partition()`, `std::__stable_partition_adaptive()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::requested_size()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

**2.28.1.32** `template<typename _Tp> void std::swap ( _Tp & __a, _Tp & __b ) [inline]`

Swaps two values.

#### Parameters

*\_\_a* A thing of arbitrary type.

*\_\_b* Another thing of arbitrary type.

#### Returns

Nothing.

Definition at line 130 of file `move.h`.

**2.28.1.33** `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator2 std::swap_ranges ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2 )`

Swap the elements of two sequences.

#### Parameters

*first1* A forward iterator.

*last1* A forward iterator.

*first2* A forward iterator.

#### Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 157 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

**2.28.1.34** `template<typename _InputIterator1, typename _InputIterator2  
, typename _OutputIterator, typename _BinaryOperation  
> _OutputIterator std::transform ( _InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator  
__result, _BinaryOperation __binary_op )`

Perform an operation on corresponding elements of two sequences.

#### Parameters

*first1* An input iterator.  
*last1* An input iterator.  
*first2* An input iterator.  
*result* An output iterator.  
*binary\_op* A binary operator.

#### Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(result+N)=binary_op(*(first1+N),*(first2+N))` for each `N` in the range `[0,last1-first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4713 of file `stl_algo.h`.

**2.28.1.35** `template<typename _InputIterator, typename _OutputIterator,  
typename _UnaryOperation > _OutputIterator std::transform ( _InputIterator  
__first, _InputIterator __last, _OutputIterator  
__result, _UnaryOperation __unary_op )`

Perform an operation on a sequence.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*result* An output iterator.  
*unary\_op* A unary operator.

#### Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates  $*(result+N)=unary\_op(*(first+N))$  for each  $N$  in the range  $[0, last-first)$ .

`unary_op` must not alter its argument.

Definition at line 4677 of file `stl_algo.h`.

**2.28.1.36** `template<typename _ForwardIterator, typename _BinaryPredicate  
> _ForwardIterator std::unique ( _ForwardIterator __first,  
_ForwardIterator __last, _BinaryPredicate __binary_pred )`

Remove consecutive values from a sequence using a predicate.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*binary\_pred* A binary predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1207 of file `stl_algo.h`.

**2.28.1.37** `template<typename _ForwardIterator> _ForwardIterator  
std::unique ( _ForwardIterator __first, _ForwardIterator __last )`

Remove consecutive duplicate values from a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is

unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1167 of file `stl_algo.h`.

**2.28.1.38** `template<typename _InputIterator, typename _OutputIterator  
> _OutputIterator std::unique_copy ( _InputIterator __first,  
_InputIterator __last, _OutputIterator __result ) [inline]`

Copy a sequence, removing consecutive duplicate values.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4877 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

**2.28.1.39** `template<typename _InputIterator, typename _OutputIterator,  
typename _BinaryPredicate > _OutputIterator std::unique_copy (`  
`_InputIterator __first, _InputIterator __last, _OutputIterator`  
`__result, _BinaryPredicate __binary_pred ) [inline]`

Copy a sequence, removing consecutive values using a predicate.

#### Parameters

*first* An input iterator.

*last* An input iterator.



**result** An output iterator.

**binary\_pred** A binary predicate.

### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements for which `binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require Copy-Constructible and Assignable?

Definition at line 4917 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

## 2.29 Non-Mutating

Collaboration diagram for Non-Mutating:



### Functions

- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`_Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`

- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

## 2.29.1 Function Documentation

### 2.29.1.1 `template<typename _ForwardIterator > _ForwardIterator` `std::adjacent_find ( _ForwardIterator __first, _ForwardIterator` `__last )`

Find two adjacent values in a sequence that are equal.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

#### Returns

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that *\*i* == *\*(i+1)*, or *last* if no such iterator exists.

Definition at line 4324 of file `stl_algo.h`.

### 2.29.1.2 `template<typename _ForwardIterator, typename _BinaryPredicate >` `_ForwardIterator std::adjacent_find ( _ForwardIterator __first,` `_ForwardIterator __last, _BinaryPredicate __binary_pred )`

Find two adjacent values in a sequence using a predicate.

**Parameters**

*first* A forward iterator.  
*last* A forward iterator.  
*binary\_pred* A binary predicate.

**Returns**

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that `binary_pred(*i,*(i+1))` is true, or *last* if no such iterator exists.

Definition at line 4356 of file `stl_algo.h`.

**2.29.1.3** `template<typename _InputIterator , typename _Predicate > bool  
 std::all_of ( _InputIterator __first, _InputIterator __last, _Predicate  
 __pred ) [inline]`

Checks that a predicate is true for all the elements of a sequence.

**Parameters**

*first* An input iterator.  
*last* An input iterator.  
*pred* A predicate.

**Returns**

True if the check is true, false otherwise.

Returns true if *pred* is true for each element in the range [*first*,*last*), and false otherwise.

Definition at line 728 of file `stl_algo.h`.

References `std::find_if_not()`.

**2.29.1.4** `template<typename _InputIterator , typename _Predicate >  
 bool std::any_of ( _InputIterator __first, _InputIterator __last,  
 _Predicate __pred ) [inline]`

Checks that a predicate is false for at least an element of a sequence.

**Parameters**

*first* An input iterator.  
*last* An input iterator.

*pred* A predicate.

### Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range [first,last) such that *pred* is true, and false otherwise.

Definition at line 762 of file `stl_algo.h`.

References `std::none_of()`.

**2.29.1.5** `template<typename _InputIterator , typename _Tp >`  
`iterator_traits<_InputIterator>::difference_type std::count (`  
`_InputIterator __first, _InputIterator __last, const _Tp & __value )`

Count the number of copies of a value in a sequence.

### Parameters

*first* An input iterator.

*last* An input iterator.

*value* The value to be counted.

### Returns

The number of iterators *i* in the range [first,last) for which `*i == value`

Definition at line 4388 of file `stl_algo.h`.

**2.29.1.6** `template<typename _InputIterator , typename _Predicate >`  
`iterator_traits<_InputIterator>::difference_type std::count_if (`  
`_InputIterator __first, _InputIterator __last, _Predicate __pred )`

Count the elements of a sequence for which a predicate is true.

### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

### Returns

The number of iterators *i* in the range [first,last) for which `pred(*i)` is true.

Definition at line 4413 of file `stl_algo.h`.

**2.29.1.7** `template<typename _II1, typename _II2 > bool std::equal ( _II1  
__first1, _II1 __last1, _II2 __first2 ) [inline]`

Tests a range for element-wise equality.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

#### Returns

A boolean true or false.

This compares the elements of two ranges using == and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1031 of file stl\_algobase.h.

Referenced by std::operator==().

**2.29.1.8** `template<typename _IIter1, typename _IIter2, typename  
_BinaryPredicate > bool std::equal ( _IIter1 __first1, _IIter1 __last1,  
_IIter2 __first2, _BinaryPredicate __binary_pred ) [inline]`

Tests a range for element-wise equality.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

*binary\_pred* A binary predicate [functor](#).

#### Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1063 of file stl\_algobase.h.

**2.29.1.9** `template<typename _InputIterator , typename _Tp > _InputIterator  
std::find ( _InputIterator __first, _InputIterator __last, const _Tp &  
__val ) [inline]`

Find the first occurrence of a value in a sequence.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*val* The value to find.

#### Returns

The first iterator *i* in the range [*first*,*last*) such that *\*i == val*, or *last* if no such iterator exists.

Definition at line 4200 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

**2.29.1.10** `template<typename _ForwardIterator1 , typename  
_ForwardIterator2 , typename _BinaryPredicate >  
_ForwardIterator1 std::find_end ( _ForwardIterator1 __first1,  
_ForwardIterator1 __last1, _ForwardIterator2 __first2,  
_ForwardIterator2 __last2, _BinaryPredicate __comp )  
[inline]`

Find last matching subsequence in a sequence using a predicate.

#### Parameters

*first1* Start of range to search.

*last1* End of range to search.

*first2* Start of sequence to match.

*last2* End of sequence to match.

*comp* The predicate to use.

#### Returns

The last iterator *i* in the range [*first1*,*last1*-(*last2*-*first2*)) such that `predicate(*(i+N), (first2+N))` is true for each *N* in the range [0,*last2*-*first2*), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1,last1-(last2-first2))`

Definition at line 694 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.29.1.11 template<typename _ForwardIterator1 , typename
 _ForwardIterator2 > _ForwardIterator1 std::find_end (
 _ForwardIterator1 __first1, _ForwardIterator1 __last1,
 _ForwardIterator2 __first2, _ForwardIterator2 __last2)
 [inline]
```

Find last matching subsequence in a sequence.

#### Parameters

*first1* Start of range to search.

*last1* End of range to search.

*first2* Start of sequence to match.

*last2* End of sequence to match.

#### Returns

The last iterator `i` in the range `[first1,last1-(last2-first2))` such that `*(i+N) == *(first2+N)` for each `N` in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1,last1-(last2-first2))`

Definition at line 647 of file `stl_algo.h`.

References `std::__iterator_category()`.



**2.29.1.12** `template<typename _InputIterator , typename _ForwardIterator ,  
 typename _BinaryPredicate > _InputIterator std::find_first_of (`  
`_InputIterator __first1, _InputIterator __last1, _ForwardIterator`  
`__first2, _ForwardIterator __last2, _BinaryPredicate __comp )`

Find element from a set in a sequence using a predicate.

#### Parameters

*first1* Start of range to search.  
*last1* End of range to search.  
*first2* Start of match candidates.  
*last2* End of match candidates.  
*comp* Predicate to use.

#### Returns

The first iterator *i* in the range [first1,last1) such that `comp(*i, *(i2))` is true and *i2* is an iterator in [first2,last2), or *last1* if no such iterator exists.

Searches the range [first1,last1) for an element that is equal to some element in the range [first2,last2). If found, returns an iterator in the range [first1,last1), otherwise returns *last1*.

Definition at line 4293 of file `stl_algo.h`.

**2.29.1.13** `template<typename _InputIterator , typename _ForwardIterator`  
`> _InputIterator std::find_first_of ( _InputIterator`  
`__first1, _InputIterator __last1, _ForwardIterator __first2,`  
`_FowardIterator __last2 )`

Find element from a set in a sequence.

#### Parameters

*first1* Start of range to search.  
*last1* End of range to search.  
*first2* Start of match candidates.  
*last2* End of match candidates.

#### Returns

The first iterator *i* in the range [first1,last1) such that `*i == *(i2)` such that *i2* is an iterator in [first2,last2), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for an element that is equal to some element in the range `[first2,last2)`. If found, returns an iterator in the range `[first1,last1)`, otherwise returns `last1`.

Definition at line 4253 of file `stl_algo.h`.

**2.29.1.14** `template<typename _InputIterator , typename _Predicate >  
_InputIterator std::find_if ( _InputIterator __first, _InputIterator  
__last, _Predicate __pred ) [inline]`

Find the first element in a sequence for which a predicate is true.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is true, or `last` if no such iterator exists.

Definition at line 4224 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

**2.29.1.15** `template<typename _InputIterator , typename _Predicate  
> _InputIterator std::find_if_not ( _InputIterator __first,  
_InputIterator __last, _Predicate __pred ) [inline]`

Find the first element in a sequence for which a predicate is false.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is false, or `last` if no such iterator exists.

Definition at line 777 of file `stl_algo.h`.

References `std::__find_if_not()`, and `std::__iterator_category()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

**2.29.1.16** `template<typename _InputIterator , typename _Function >  
 _Function std::for_each ( _InputIterator __first, _InputIterator  
 __last, _Function __f )`

Apply a function to every element of a sequence.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*f* A unary function object.

#### Returns

*f* (std::move(*f*) in C++0x).

Applies the function object *f* to each element in the range [first,last). *f* must not modify the order of the sequence. If *f* has a return value it is ignored.

Definition at line 4179 of file stl\_algo.h.

**2.29.1.17** `template<typename _InputIterator1 , typename _InputIterator2  
 , typename _BinaryPredicate > pair<_InputIterator1,  
 _InputIterator2> std::mismatch ( _InputIterator1 __first1,  
 _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate  
 __binary_pred )`

Finds the places in ranges which don't match.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

*binary\_pred* A binary predicate [functor](#).

#### Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the *binary\_pred* parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1206 of file stl\_algobase.h.

**2.29.1.18** `template<typename _InputIterator1, typename _InputIterator2  
> pair<_InputIterator1, _InputIterator2> std::mismatch (   
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
__first2 )`

Finds the places in ranges which don't match.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

#### Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using == and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1168 of file stl\_algobase.h.

**2.29.1.19** `template<typename _InputIterator, typename _Predicate > bool  
std::none_of ( _InputIterator __first, _InputIterator __last,  
_Predicate __pred ) [inline]`

Checks that a predicate is false for all the elements of a sequence.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

True if the check is true, false otherwise.

Returns true if *pred* is false for each element in the range [first,last), and false otherwise.

Definition at line 745 of file stl\_algo.h.

Referenced by std::any\_of(), and std::is\_partitioned().

**2.29.1.20** `template<typename _ForwardIterator1, typename  
_ForwardIterator2 > _ForwardIterator1 std::search (  
_ForwardIterator1 __first1, _ForwardIterator1 __last1,  
_ForwardIterator2 __first2, _ForwardIterator2 __last2 )`

Search a sequence for a matching sub-sequence.

#### Parameters

*first1* A forward iterator.

*last1* A forward iterator.

*first2* A forward iterator.

*last2* A forward iterator.

#### Returns

The first iterator *i* in the range  $[first1, last1 - (last2 - first2))$  such that  $*(i+N) == *(first2+N)$  for each *N* in the range  $[0, last2 - first2)$ , or *last1* if no such iterator exists.

Searches the range  $[first1, last1)$  for a sub-sequence that compares equal value-by-value with the sequence given by  $[first2, last2)$  and returns an iterator to the first element of the sub-sequence, or *last1* if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range  $[first1, last1)$  it must start at a position less than  $last1 - (last2 - first2)$  where  $last2 - first2$  is the length of the sub-sequence. This means that the returned iterator *i* will be in the range  $[first1, last1 - (last2 - first2))$

Definition at line 4453 of file `stl_algo.h`.

**2.29.1.21** `template<typename _ForwardIterator1, typename  
_ForwardIterator2, typename _BinaryPredicate >  
_ForwardIterator1 std::search ( _ForwardIterator1 __first1,  
_ForwardIterator1 __last1, _ForwardIterator2 __first2,  
_ForwardIterator2 __last2, _BinaryPredicate __predicate )`

Search a sequence for a matching sub-sequence using a predicate.

#### Parameters

*first1* A forward iterator.

*last1* A forward iterator.

*first2* A forward iterator.

*last2* A forward iterator.

*predicate* A binary predicate.

### Returns

The first iterator *i* in the range `[first1,last1-(last2-first2))` such that `predicate(*(i+N),*(first2+N))` is true for each *N* in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)`, using `predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `last1` if no such iterator exists.

### See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4525 of file `stl_algo.h`.

**2.29.1.22** `template<typename _ForwardIterator, typename _Integer, typename _Tp> _ForwardIterator std::search_n ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val )`

Search a sequence for a number of consecutive values.

### Parameters

*first* A forward iterator.

*last* A forward iterator.

*count* The number of consecutive values.

*val* The value to find.

### Returns

The first iterator *i* in the range `[first,last-count)` such that `*(i+N) == val` for each *N* in the range `[0,count)`, or `last` if no such iterator exists.

Searches the range `[first,last)` for `count` consecutive elements equal to `val`.

Definition at line 4598 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

**2.29.1.23** `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate > _ForwardIterator  
std::search_n ( _ForwardIterator __first, _ForwardIterator  
__last, _Integer __count, const _Tp & __val, _BinaryPredicate  
__binary_pred )`

Search a sequence for a number of consecutive values using a predicate.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*count* The number of consecutive values.

*val* The value to find.

*binary\_pred* A binary predicate.

#### Returns

The first iterator *i* in the range `[first,last-count)` such that `binary_pred(*(i+N),val)` is true for each *N* in the range `[0,count)`, or *last* if no such iterator exists.

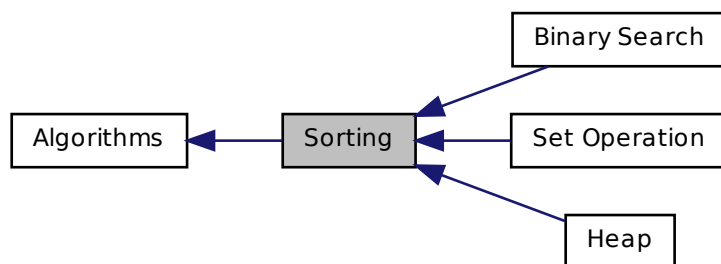
Searches the range `[first,last)` for `count` consecutive elements for which the predicate returns true.

Definition at line 4635 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

## 2.30 Sorting

Collaboration diagram for Sorting:



### Modules

- [Set Operation](#)
- [Binary Search](#)
- [Heap](#)

### Functions

- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`



- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`  
`__last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`  
`__last2, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _`  
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _`  
`Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`  
`&__b, _Compare __comp)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`  
`&__b)`

- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_-`  
`ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_-`  
`ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __-`  
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`  
`RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __-`  
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`  
`RandomAccessIterator __result_last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`

### 2.30.1 Function Documentation

#### 2.30.1.1 `template<typename _BidirectionalIterator > void std::inplace_merge` `( _BidirectionalIterator __first, _BidirectionalIterator __middle,` `_BidirectionalIterator __last )`

Merges two sorted ranges in place.

##### Parameters

*first* An iterator.

*middle* Another iterator.

*last* Another iterator.

##### Returns

Nothing.

Merges two sorted and consecutive ranges, `[first,middle)` and `[middle,last)`, and puts the result in `[first,last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes  $(last-first)-1$  comparisons. Otherwise an  $N\log N$  algorithm is used, where  $N$  is `distance(first,last)`.

Definition at line 3061 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, `std::distance()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

#### 2.30.1.2 `template<typename _BidirectionalIterator, typename _Compare` `> void std::inplace_merge ( _BidirectionalIterator __first,` `_BidirectionalIterator __middle, _BidirectionalIterator __last,` `_Compare __comp )`

Merges two sorted ranges in place.

**Parameters**

*first* An iterator.

*middle* Another iterator.

*last* Another iterator.

*comp* A functor to use for comparisons.

**Returns**

Nothing.

Merges two sorted and consecutive ranges, `[first,middle)` and `[middle,last)`, and puts the result in `[first,last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes  $(last-first)-1$  comparisons. Otherwise an  $N\log N$  algorithm is used, where  $N$  is `distance(first,last)`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 3116 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::distance()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

### 2.30.1.3 `template<typename _ForwardIterator> bool std::is_sorted (` `_ForwardIterator __first, _ForwardIterator __last ) [inline]`

Determines whether the elements of a sequence are sorted.

**Parameters**

*first* An iterator.

*last* Another iterator.

**Returns**

True if the elements are sorted, false otherwise.

Definition at line 3816 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**2.30.1.4** `template<typename _ForwardIterator, typename _Compare> bool  
std::is_sorted ( _ForwardIterator __first, _ForwardIterator __last,  
_Compare __comp ) [inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

#### Parameters

*first* An iterator.

*last* Another iterator.

*comp* A comparison functor.

#### Returns

True if the elements are sorted, false otherwise.

Definition at line 3830 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**2.30.1.5** `template<typename _ForwardIterator> _ForwardIterator  
std::is_sorted_until ( _ForwardIterator __first, _ForwardIterator  
__last )`

Determines the end of a sorted sequence.

#### Parameters

*first* An iterator.

*last* Another iterator.

#### Returns

An iterator pointing to the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is sorted.

Definition at line 3844 of file `stl_algo.h`.

**2.30.1.6** `template<typename _ForwardIterator, typename _Compare>  
_ForwardIterator std::is_sorted_until ( _ForwardIterator __first,  
_ForwardIterator __last, _Compare __comp )`

Determines the end of a sorted sequence using comparison functor.

**Parameters**

*first* An iterator.  
*last* Another iterator.  
*comp* A comparison functor.

**Returns**

An iterator pointing to the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is sorted.

Definition at line 3873 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

**2.30.1.7** `template<typename _II1 , typename _II2 > bool  
std::lexicographical_compare ( _II1 __first1, _II1 __last1, _II2  
__first2, _II2 __last2 ) [inline]`

Performs **dictionary** comparison on ranges.

**Parameters**

*first1* An input iterator.  
*last1* An input iterator.  
*first2* An input iterator.  
*last2* An input iterator.

**Returns**

A boolean true or false.

*Returns true if the sequence of elements defined by the range [*first1*,*last1*) is lexicographically less than the sequence of elements defined by the range [*first2*,*last2*). Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1094 of file `stl_algobase.h`.

**2.30.1.8** `template<typename _II1 , typename _II2 , typename _Compare >  
bool std::lexicographical_compare ( _II1 __first1, _II1 __last1, _II2  
__first2, _II2 __last2, _Compare __comp )`

Performs **dictionary** comparison on ranges.

**Parameters**

*first1* An input iterator.  
*last1* An input iterator.  
*first2* An input iterator.  
*last2* An input iterator.  
*comp* A [comparison functor](#).

**Returns**

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1128 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

**2.30.1.9** `template<typename _Tp> const _Tp & std::max ( const _Tp & __a,  
const _Tp & __b ) [inline]`

This does what you think it does.

**Parameters**

*a* A thing of arbitrary type.  
*b* Another thing of arbitrary type.

**Returns**

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 208 of file `stl_algobase.h`.

Referenced by `std::Deque_base< _Tp, _Alloc >::_M_initialize_map()`, and `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

**2.30.1.10** `template<typename _Tp, typename _Compare> const _Tp &  
std::max ( const _Tp & __a, const _Tp & __b, _Compare __comp  
) [inline]`

This does what you think it does.

**Parameters**

*a* A thing of arbitrary type.  
*b* Another thing of arbitrary type.  
*comp* A [comparison functor](#).

**Returns**

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 252 of file `stl_algobase.h`.

**2.30.1.11** `template<typename _ForwardIterator > _ForwardIterator  
 std::max_element ( _ForwardIterator __first, _ForwardIterator  
 __last )`

Return the maximum element in a range.

**Parameters**

*first* Start of range.  
*last* End of range.

**Returns**

Iterator referencing the first instance of the largest value.

Definition at line 6035 of file `stl_algo.h`.

**2.30.1.12** `template<typename _ForwardIterator , typename _Compare >  
 _ForwardIterator std::max_element ( _ForwardIterator __first,  
 _ForwardIterator __last, _Compare __comp )`

Return the maximum element in a range using comparison functor.

**Parameters**

*first* Start of range.  
*last* End of range.  
*comp* Comparison functor.

**Returns**

Iterator referencing the first instance of the largest value according to *comp*.



Definition at line 6063 of file stl\_algo.h.

Referenced by std::valarray< \_Tp >::max().

**2.30.1.13** `template<typename _InputIterator1 , typename _InputIterator2  
, typename _OutputIterator > _OutputIterator std::merge (  
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
__first2, _InputIterator2 __last2, _OutputIterator __result )`

Merges two sorted ranges.

#### Parameters

*first1* An iterator.

*first2* Another iterator.

*last1* Another iterator.

*last2* Another iterator.

*result* An iterator pointing to the end of the merged range.

#### Returns

An iterator pointing to the first element *not less than val*.

Merges the ranges [first1,last1) and [first2,last2) into the sorted range [result, result + (last1-first1) + (last2-first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5278 of file stl\_algo.h.

**2.30.1.14** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator , typename _Compare > _OutputIterator  
std::merge ( _InputIterator1 __first1, _InputIterator1 __last1,  
_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator  
__result, _Compare __comp )`

Merges two sorted ranges.

#### Parameters

*first1* An iterator.

*first2* Another iterator.

*last1* Another iterator.

*last2* Another iterator.

*result* An iterator pointing to the end of the merged range.

*comp* A functor to use for comparisons.

### Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges [first1,last1) and [first2,last2) into the sorted range [result, result + (last1-first1) + (last2-first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5341 of file stl\_algo.h.

**2.30.1.15** `template<typename _Tp, typename _Compare> const _Tp &  
std::min ( const _Tp & __a, const _Tp & __b, _Compare __comp )  
[inline]`

This does what you think it does.

### Parameters

*a* A thing of arbitrary type.

*b* Another thing of arbitrary type.

*comp* A [comparison functor](#).

### Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 231 of file stl\_algobase.h.

**2.30.1.16** `template<typename _Tp> const _Tp & std::min ( const _Tp & __a,  
const _Tp & __b ) [inline]`

This does what you think it does.

**Parameters**

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.

**Returns**

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 185 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_profile::__report()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::generate_canonical()`, `__gnu_cxx::random_sample_n()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`, `std::basic_string< _CharT, _Traits, _Alloc >::rfind()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**2.30.1.17** `template<typename _ForwardIterator > _ForwardIterator  
std::min_element ( _ForwardIterator __first, _ForwardIterator  
__last )`

Return the minimum element in a range.

**Parameters**

- first* Start of range.
- last* End of range.

**Returns**

Iterator referencing the first instance of the smallest value.

Definition at line 5979 of file `stl_algo.h`.

**2.30.1.18** `template<typename _ForwardIterator, typename _Compare >  
_ForwardIterator std::min_element ( _ForwardIterator __first,  
_ForwardIterator __last, _Compare __comp )`

Return the minimum element in a range using comparison functor.

**Parameters**

*first* Start of range.  
*last* End of range.  
*comp* Comparison functor.

**Returns**

Iterator referencing the first instance of the smallest value according to comp.

Definition at line 6007 of file stl\_algo.h.

Referenced by std::valarray<\_Tp>::min().

**2.30.1.19** `template<typename _Tp , typename _Compare > pair< const _Tp  
&, const _Tp & > std::minmax ( const _Tp & __a, const _Tp &  
__b, _Compare __comp ) [inline]`

Determines min and max at once as an ordered pair.

**Parameters**

*a* A thing of arbitrary type.  
*b* Another thing of arbitrary type.  
*comp* A [comparison functor](#).

**Returns**

A pair(b, a) if b is smaller than a, pair(a, b) otherwise.

Definition at line 3921 of file stl\_algo.h.

**2.30.1.20** `template<typename _Tp > pair< const _Tp &, const _Tp & >  
std::minmax ( const _Tp & __a, const _Tp & __b ) [inline]`

Determines min and max at once as an ordered pair.

**Parameters**

*a* A thing of arbitrary type.  
*b* Another thing of arbitrary type.

**Returns**

A pair(b, a) if b is smaller than a, pair(a, b) otherwise.

Definition at line 3902 of file stl\_algo.h.

**2.30.1.21** `template<typename _ForwardIterator > pair<_ForwardIterator,  
_ForwardIterator> std::minmax_element ( _ForwardIterator  
__first, _ForwardIterator __last )`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

#### Parameters

*first* Start of range.

*last* End of range.

#### Returns

make\_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 3940 of file stl\_algo.h.

References std::make\_pair().

**2.30.1.22** `template<typename _ForwardIterator , typename _Compare >  
pair<_ForwardIterator, _ForwardIterator> std::minmax_element (   
_ForwardIterator __first, _ForwardIterator __last, _Compare  
__comp )`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

#### Parameters

*first* Start of range.

*last* End of range.

*comp* Comparison functor.

#### Returns

make\_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 4016 of file stl\_algo.h.

References std::make\_pair().

**2.30.1.23** `template<typename _BidirectionalIterator > bool  
std::next_permutation ( _BidirectionalIterator __first,  
_BidirectionalIterator __last )`

Permute range into the next *dictionary* ordering.

#### Parameters

*first* Start of range.

*last* End of range.

#### Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3518 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.30.1.24** `template<typename _BidirectionalIterator , typename _Compare  
> bool std::next_permutation ( _BidirectionalIterator __first,  
_BidirectionalIterator __last, _Compare __comp )`

Permute range into the next *dictionary* ordering using comparison functor.

#### Parameters

*first* Start of range.

*last* End of range.

*comp* A comparison functor.

#### Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range `[first,last)` as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3575 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.30.1.25** `template<typename _RandomAccessIterator > void std::nth_element  
( _RandomAccessIterator __first, _RandomAccessIterator __nth,  
_RandomAccessIterator __last ) [inline]`

Sort a sequence just enough to find a particular position.

#### Parameters

*first* An iterator.

*nth* Another iterator.

*last* Another iterator.

#### Returns

Nothing.

Rearranges the elements in the range `[first,last)` so that `*nth` is the same element that would have been in that position had the whole sequence been sorted. whole sequence been sorted. The elements either side of `*nth` are not completely sorted, but for any iterator in the range `[first,nth)` and any iterator in the range `[nth,last)` it holds that `*j < *i` is false.

Definition at line 5123 of file `stl_algo.h`.

References `std::__lg()`.

**2.30.1.26** `template<typename _RandomAccessIterator , typename _Compare  
> void std::nth_element ( _RandomAccessIterator __first,  
_RandomAccessIterator __nth, _RandomAccessIterator __last,  
_Compare __comp ) [inline]`

Sort a sequence just enough to find a particular position using a predicate for comparison.

#### Parameters

*first* An iterator.

*nth* Another iterator.

*last* Another iterator.

*comp* A comparison functor.

#### Returns

Nothing.

Rearranges the elements in the range `[first,last)` so that `*nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*nth` are not completely sorted, but for any iterator in the range `[first,nth)` and any iterator in the range `[nth,last)` it holds that `comp(*j,*i)` is false.

Definition at line 5162 of file `stl_algo.h`.

References `std::__lg()`.

**2.30.1.27** `template<typename _RandomAccessIterator > void std::partial_sort`  
`( _RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last ) [inline]`

Sort the smallest elements of a sequence.

#### Parameters

*first* An iterator.

*middle* Another iterator.

*last* Another iterator.

#### Returns

Nothing.

Sorts the smallest (middle-first) elements in the range `[first,last)` and moves them to the range `[first,middle)`. The order of the remaining elements in the range `[middle,last)` is undefined. After the sort if `i` and `j` are iterators in the range `[first,middle)` such that `i` precedes `j` and `j` is an iterator in the range `[middle,last)` then `*j < *i` and `*k < *i` are both false.

Definition at line 5046 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

**2.30.1.28** `template<typename _RandomAccessIterator, typename _Compare`  
`> void std::partial_sort ( _RandomAccessIterator __first,`  
`_RandomAccessIterator __middle, _RandomAccessIterator __last,`  
`_Compare __comp ) [inline]`

Sort the smallest elements of a sequence using a predicate for comparison.

#### Parameters

*first* An iterator.

*middle* Another iterator.



*last* Another iterator.

*comp* A comparison functor.

### Returns

Nothing.

Sorts the smallest (middle-first) elements in the range `[first,last)` and moves them to the range `[first,middle)`. The order of the remaining elements in the range `[middle,last)` is undefined. After the sort if `i` and `are` are iterators in the range `[first,middle)` such that `precedes` and `is an iterator` in the range `[middle,last)` then `*comp(j,*i)` and `comp(*k,*i)` are both false.

Definition at line 5085 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

**2.30.1.29** `template<typename _InputIterator , typename  
_RandomAccessIterator , typename _Compare >  
_RandomAccessIterator std::partial_sort_copy ( _InputIterator  
__first, _InputIterator __last, _RandomAccessIterator __result_first,  
_RandomAccessIterator __result_last, _Compare __comp )`

Copy the smallest elements of a sequence using a predicate for comparison.

### Parameters

*first* An input iterator.

*last* Another input iterator.

*result\_first* A random-access iterator.

*result\_last* Another random-access iterator.

*comp* A comparison functor.

### Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest `N` values from the range `[first,last)` to the range beginning at `result_first`, where the number of elements to be copied, `N`, is the smaller of `(last-first)` and `(result_last-result_first)`. After the sort if `i` and `are` are iterators in the range `[result_first,result_first+N)` such that `precedes` then `comp(*j,*i)` is false. The value returned is `result_first+N`.

Definition at line 2006 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

**2.30.1.30** `template<typename _InputIterator , typename  
_RandomAccessIterator > _RandomAccessIterator  
std::partial_sort_copy ( _InputIterator __first,  
_InputIterator __last, _RandomAccessIterator __result_first,  
_RandomAccessIterator __result_last )`

Copy the smallest elements of a sequence.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*result\_first* A random-access iterator.  
*result\_last* Another random-access iterator.

#### Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest  $N$  values from the range  $[first, last)$  to the range beginning at `result_first`, where the number of elements to be copied,  $N$ , is the smaller of  $(last - first)$  and  $(result\_last - result\_first)$ . After the sort if  $i$  and  $j$  are iterators in the range  $[result\_first, result\_first + N)$  such that  $j$  precedes  $i$  then  $*j < *i$  is false. The value returned is `result_first + N`.

Definition at line 1940 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

**2.30.1.31** `template<typename _BidirectionalIterator , typename _Compare  
> bool std::prev_permutation ( _BidirectionalIterator __first,  
_BidirectionalIterator __last, _Compare __comp )`

Permute range into the previous *dictionary* ordering using comparison functor.

#### Parameters

*first* Start of range.  
*last* End of range.  
*comp* A comparison functor.

#### Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range `[first,last)` as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3688 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.30.1.32** `template<typename _BidirectionalIterator > bool  
std::prev_permutation ( _BidirectionalIterator __first,  
_BidirectionalIterator __last )`

Permute range into the previous *dictionary* ordering.

#### Parameters

*first* Start of range.

*last* End of range.

#### Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3631 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

**2.30.1.33** `template<typename _RandomAccessIterator , typename  
_Compare > void std::sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp ) [inline]`

Sort the elements of a sequence using a predicate for comparison.

#### Parameters

*first* An iterator.

*last* Another iterator.

*comp* A comparison functor.

#### Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for every iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5236 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

**2.30.1.34** `template<typename _RandomAccessIterator> void std::sort (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`  
`[inline]`

Sort the elements of a sequence.

#### Parameters

*first* An iterator.

*last* Another iterator.

#### Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5200 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

**2.30.1.35** `template<typename _RandomAccessIterator , typename _Compare`  
`> void std::stable_sort ( _RandomAccessIterator __first,`  
`_RandomAccessIterator __last, _Compare __comp ) [inline]`

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

#### Parameters

*first* An iterator.

*last* Another iterator.

*comp* A comparison functor.

**Returns**

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `comp(x,y)` is false and `comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5442 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

```
2.30.1.36 template<typename _RandomAccessIterator > void std::stable_sort
 (_RandomAccessIterator __first, _RandomAccessIterator __last)
 [inline]
```

Sort the elements of a sequence, preserving the relative order of equivalent elements.

**Parameters**

*first* An iterator.

*last* Another iterator.

**Returns**

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `x<y` is false and `y<x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5400 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

## 2.31 Set Operation

Collaboration diagram for Set Operation:



### Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`  
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`  
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`  
`result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result)`

- `template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , type-`  
`name _Compare >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`

### 2.31.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

### 2.31.2 Function Documentation

**2.31.2.1** `template<typename _InputIterator1 , typename _InputIterator2 >`  
`bool std::includes ( _InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2 )`

Determines whether all elements of a sequence exists in a range.

#### Parameters

*first1* Start of search range.  
*last1* End of search range.  
*first2* Start of sequence  
*last2* End of sequence.

#### Returns

True if each element in [first2,last2) is contained in order within [first1,last1). False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1). The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3414 of file `stl_algo.h`.

**2.31.2.2** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _Compare > bool std::includes ( _InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _Compare __comp )`

Determines whether all elements of a sequence exists in a range using comparison.

#### Parameters

*first1* Start of search range.  
*last1* End of search range.  
*first2* Start of sequence  
*last2* End of sequence.  
*comp* Comparison function to use.

#### Returns

True if each element in [first2,last2) is contained in order within [first1,last1) according to comp. False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1), using comp to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3464 of file stl\_algo.h.

**2.31.2.3** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator , typename _Compare > _OutputIterator  
std::set_difference ( _InputIterator1 __first1, _InputIterator1  
__last1, _InputIterator2 __first2, _InputIterator2 __last2,  
_OutputIterator __result, _Compare __comp )`

Return the difference of two sorted ranges using comparison functor.

#### Parameters

*first1* Start of first range.  
*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.  
*comp* The comparison functor.



**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to *comp*, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5799 of file `stl_algo.h`.

**2.31.2.4** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator > _OutputIterator std::set_difference (  
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
__first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the difference of two sorted ranges.

**Parameters**

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5738 of file `stl_algo.h`.

**2.31.2.5** `template<typename _InputIterator1 , typename _InputIterator2 ,  
 typename _OutputIterator > _OutputIterator std::set_intersection (  
 _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
 __first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the intersection of two sorted ranges.

#### Parameters

*first1* Start of first range.  
*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5623 of file `stl_algo.h`.

**2.31.2.6** `template<typename _InputIterator1 , typename _InputIterator2 ,  
 typename _OutputIterator , typename _Compare > _OutputIterator  
 std::set_intersection ( _InputIterator1 __first1, _InputIterator1  
 __last1, _InputIterator2 __first2, _InputIterator2 __last2,  
 _OutputIterator __result, _Compare __comp )`

Return the intersection of two sorted ranges using comparison functor.

#### Parameters

*first1* Start of first range.  
*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.  
*comp* The comparison functor.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that iterator advances. If an element is contained in both ranges according to *comp*, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5680 of file `stl_algo.h`.

**2.31.2.7** `template<typename _InputIterator1 , typename  
_InputIterator2 , typename _OutputIterator > _OutputIterator  
std::set_symmetric_difference ( _InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _OutputIterator __result )`

Return the symmetric difference of two sorted ranges.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5857 of file `stl_algo.h`.

**2.31.2.8** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator , typename _Compare > _OutputIterator  
std::set_symmetric_difference ( _InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _OutputIterator __result, _Compare __comp )`

Return the symmetric difference of two sorted ranges using comparison functor.

#### Parameters

*first1* Start of first range.

*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.  
*comp* The comparison functor.

### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5923 of file `stl_algo.h`.

**2.31.2.9** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator > _OutputIterator std::set_union (`  
`_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the union of two sorted ranges.

### Parameters

*first1* Start of first range.  
*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.

### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5489 of file `stl_algo.h`.

**2.31.2.10** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator , typename _Compare > _OutputIterator  
std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1,  
_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator  
__result, _Compare __comp )`

Return the union of two sorted ranges using a comparison functor.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

*comp* The comparison functor.

#### Returns

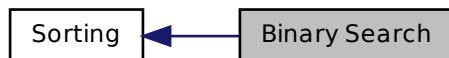
End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advanced. If an equivalent element according to *comp* is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5556 of file `stl_algo.h`.

## 2.32 Binary Search

Collaboration diagram for Binary Search:



## Functions

- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val)`

### 2.32.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal\_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions

the range.

### 2.32.2 Function Documentation

**2.32.2.1** `template<typename _ForwardIterator, typename _Tp> bool  
std::binary_search ( _ForwardIterator __first, _ForwardIterator  
__last, const _Tp & __val )`

Determines whether an element exists in a range.

#### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

#### Returns

True if *val* (or its equivalent) is in [*first*,*last* ].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2668 of file `stl_algo.h`.

References `std::lower_bound()`.

**2.32.2.2** `template<typename _ForwardIterator, typename _Tp, typename  
_Compare> bool std::binary_search ( _ForwardIterator __first,  
_ForwardIterator __last, const _Tp & __val, _Compare __comp )`

Determines whether an element exists in a range.

#### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

*comp* A functor to use for comparisons.

#### Returns

True if *val* (or its equivalent) is in [*first*,*last* ].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2701 of file `stl_algo.h`.

References `std::lower_bound()`.

**2.32.2.3** `template<typename _ForwardIterator, typename _Tp, typename  
_Compare > pair<_ForwardIterator, _ForwardIterator>  
std::equal_range ( _ForwardIterator __first, _ForwardIterator  
__last, const _Tp & __val, _Compare __comp )`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*val* The search term.  
*comp* A functor to use for comparisons.

#### Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val, comp),
 upper_bound(first, last, val, comp))
```

but does not actually call those functions.

Definition at line 2607 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

**2.32.2.4** `template<typename _ForwardIterator, typename _Tp >  
pair<_ForwardIterator, _ForwardIterator> std::equal_range (   
_ForwardIterator __first, _ForwardIterator __last, const _Tp &  
__val )`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.



**Parameters**

*first* An iterator.

*last* Another iterator.

*val* The search term.

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val),
 upper_bound(first, last, val))
```

but does not actually call those functions.

Definition at line 2545 of file stl\_algo.h.

References std::advance(), std::distance(), std::lower\_bound(), and std::upper\_bound().

**2.32.2.5** `template<typename _ForwardIterator, typename _Tp, typename  
_Compare> _ForwardIterator std::lower_bound ( _ForwardIterator  
__first, _ForwardIterator __last, const _Tp & __val, _Compare  
__comp )`

Finds the first position in which *val* could be inserted without changing the ordering.

**Parameters**

*first* An iterator.

*last* Another iterator.

*val* The search term.

*comp* A functor to use for comparisons.

**Returns**

An iterator pointing to the first element *not less than val*, or end() if every element is less than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2390 of file stl\_algo.h.

References std::advance(), and std::distance().

**2.32.2.6** `template<typename _ForwardIterator, typename _Tp >  
 _ForwardIterator std::lower_bound ( _ForwardIterator __first,  
 _ForwardIterator __last, const _Tp & __val )`

Finds the first position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

#### Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

Definition at line 955 of file `stl_algobase.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

**2.32.2.7** `template<typename _ForwardIterator, typename _Tp, typename  
 _Compare > _ForwardIterator std::upper_bound ( _ForwardIterator  
 __first, _ForwardIterator __last, const _Tp & __val, _Compare  
 __comp )`

Finds the last position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

*comp* A functor to use for comparisons.

#### Returns

An iterator pointing to the first element greater than *val*, or `end()` if no elements are greater than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2490 of file stl\_algo.h.

References std::advance(), and std::distance().

Referenced by std::\_\_merge\_adaptive(), std::\_\_merge\_without\_buffer(), and std::equal\_range().

**2.32.2.8** `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator std::upper_bound ( _ForwardIterator __first,  
_ForwardIterator __last, const _Tp & __val )`

Finds the last position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

#### Returns

An iterator pointing to the first element greater than *val*, or end() if no elements are greater than *val*.

Definition at line 2439 of file stl\_algo.h.

References std::advance(), and std::distance().

## 2.33 Allocators

Collaboration diagram for Allocators:



#### Classes

- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`

*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).*

*Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*

- `class __gnu_cxx::__pool_alloc< _Tp >`  
*Allocator using a memory pool with a single lock.*
- `class __gnu_cxx::__ExtPtr_allocator< _Tp >`  
*An example allocator which uses a non-standard pointer type.  
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*
- `class __gnu_cxx::array_allocator< _Tp, _Array >`  
*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- `class __gnu_cxx::bitmap_allocator< _Tp >`  
*Bitmap Allocator, primary template.*
- `class __gnu_cxx::debug_allocator< _Alloc >`  
*A meta-allocator with debugging bits, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator new
  - all deallocation calls operator delete.
- `class __gnu_cxx::malloc_allocator< _Tp >`  
*An allocator that uses malloc.  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls malloc
  - all deallocation calls free.
- `class __gnu_cxx::new_allocator< _Tp >`  
*An allocator that uses global new, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator new
  - all deallocation calls operator delete.
- `class __gnu_cxx::throw_allocator_base< _Tp, _Cond >`  
*Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.  
Note: Deallocate not allowed to throw.*

- class `std::allocator<_Tp>`

*The standard allocator, as per [20.4].*

*Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*

### 2.33.1 Detailed Description

Classes encapsulating memory operations.

## 2.34 Atomics

### Classes

- struct `std::atomic<_Tp>`

*atomic /// 29.4.3, Generic atomic type, primary class template.*

- struct `std::atomic<_Tp*>`

*Partial specialization for pointer types.*

- struct `std::atomic<bool>`

*Explicit specialization for bool.*

- struct `std::atomic<char>`

*Explicit specialization for char.*

- struct `std::atomic<char16_t>`

*Explicit specialization for char16\_t.*

- struct `std::atomic<char32_t>`

*Explicit specialization for char32\_t.*

- struct `std::atomic<int>`

*Explicit specialization for int.*

- struct `std::atomic<long>`

*Explicit specialization for long.*

- struct `std::atomic<long long>`

*Explicit specialization for long long.*

- struct `std::atomic<short>`

*Explicit specialization for short.*

- struct `std::atomic< signed char >`

*Explicit specialization for signed char.*

- struct `std::atomic< unsigned char >`

*Explicit specialization for unsigned char.*

- struct `std::atomic< unsigned int >`

*Explicit specialization for unsigned int.*

- struct `std::atomic< unsigned long >`

*Explicit specialization for unsigned long.*

- struct `std::atomic< unsigned long long >`

*Explicit specialization for unsigned long long.*

- struct `std::atomic< unsigned short >`

*Explicit specialization for unsigned short.*

- struct `std::atomic< void * >`

*Explicit specialization for void\*.*

- struct `std::atomic< wchar_t >`

*Explicit specialization for wchar\_t.*

## Defines

- #define `_ATOMIC_CMPEXCHNG__(__a, __e, __m, __x)`
- #define `_ATOMIC_LOAD__(__a, __x)`
- #define `_ATOMIC_MODIFY__(__a, __o, __m, __x)`
- #define `_ATOMIC_STORE__(__a, __m, __x)`
- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

## Typedefs

- typedef struct std::\_\_atomic\_flag\_base **std::\_\_atomic\_flag\_base**
- typedef \_\_atomic\_base< char > [atomic\\_char](#)
- typedef \_\_atomic\_base< char16\_t > [atomic\\_char16\\_t](#)
- typedef \_\_atomic\_base< char32\_t > [atomic\\_char32\\_t](#)
- typedef \_\_atomic\_base< int > [atomic\\_int](#)
- typedef [atomic\\_short](#) **std::atomic\_int\_fast16\_t**
- typedef [atomic\\_int](#) **std::atomic\_int\_fast32\_t**
- typedef [atomic\\_llong](#) **std::atomic\_int\_fast64\_t**
- typedef [atomic\\_schar](#) **std::atomic\_int\_fast8\_t**
- typedef [atomic\\_short](#) **std::atomic\_int\_least16\_t**
- typedef [atomic\\_int](#) **std::atomic\_int\_least32\_t**
- typedef [atomic\\_llong](#) **std::atomic\_int\_least64\_t**
- typedef [atomic\\_schar](#) **std::atomic\_int\_least8\_t**
- typedef [atomic\\_llong](#) **std::atomic\_intmax\_t**
- typedef [atomic\\_long](#) **std::atomic\_intptr\_t**
- typedef \_\_atomic\_base< long long > [atomic\\_llong](#)
- typedef \_\_atomic\_base< long > [atomic\\_long](#)
- typedef [atomic\\_long](#) **std::atomic\_ptrdiff\_t**
- typedef \_\_atomic\_base< signed char > [atomic\\_schar](#)
- typedef \_\_atomic\_base< short > [atomic\\_short](#)
- typedef [atomic\\_ulong](#) **std::atomic\_size\_t**
- typedef [atomic\\_long](#) **std::atomic\_ssize\_t**
- typedef \_\_atomic\_base< unsigned char > [atomic\\_uchar](#)
- typedef \_\_atomic\_base< unsigned int > [atomic\\_uint](#)
- typedef [atomic\\_ushort](#) **std::atomic\_uint\_fast16\_t**
- typedef [atomic\\_uint](#) **std::atomic\_uint\_fast32\_t**
- typedef [atomic\\_ullong](#) **std::atomic\_uint\_fast64\_t**
- typedef [atomic\\_uchar](#) **std::atomic\_uint\_fast8\_t**
- typedef [atomic\\_ushort](#) **std::atomic\_uint\_least16\_t**
- typedef [atomic\\_uint](#) **std::atomic\_uint\_least32\_t**
- typedef [atomic\\_ullong](#) **std::atomic\_uint\_least64\_t**
- typedef [atomic\\_uchar](#) **std::atomic\_uint\_least8\_t**
- typedef [atomic\\_ullong](#) **std::atomic\_uintmax\_t**
- typedef [atomic\\_ulong](#) **std::atomic\_uintptr\_t**
- typedef \_\_atomic\_base< unsigned long long > [atomic\\_ullong](#)
- typedef \_\_atomic\_base< unsigned long > [atomic\\_ulong](#)
- typedef \_\_atomic\_base< unsigned short > [atomic\\_ushort](#)
- typedef \_\_atomic\_base< wchar\_t > [atomic\\_wchar\\_t](#)
- typedef enum [std::memory\\_order](#) **std::memory\_order**

## Enumerations

- enum `std::memory_order` {  
     `memory_order_relaxed`,   `memory_order_consume`,   `memory_order_-`  
     `acquire`, `memory_order_release`,  
     `memory_order_acq_rel`, `memory_order_seq_cst` }

## Functions

- void `std::__atomic_flag_wait_explicit` (\_\_atomic\_flag\_base \*, memory\_order)  
     \_GLIBCXX\_NOTHROW
- `std::__attribute__` ((\_\_const\_\_) \_\_atomic\_flag\_base \* \_\_atomic\_flag\_for\_  
     address(const void \*\_\_z) \_GLIBCXX\_NOTHROW
- memory\_order `std::calculate_memory_order` (memory\_order \_\_m)
- template<typename \_ITp >  
     bool `std::atomic_compare_exchange_strong` (\_\_atomic\_base< \_ITp > \*\_\_a,  
     \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool `std::atomic_compare_exchange_strong` (atomic\_address \*\_\_a, void \*\*\_  
     \_\_v1, void \*\_\_v2)
- bool `std::atomic_compare_exchange_strong` (atomic\_bool \*\_\_a, bool \*\_\_i1,  
     bool \_\_i2)
- template<typename \_ITp >  
     bool `std::atomic_compare_exchange_strong_explicit` (\_\_atomic\_base< \_ITp  
     > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `std::atomic_compare_exchange_strong_explicit` (atomic\_address \*\_\_a,  
     void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `std::atomic_compare_exchange_strong_explicit` (atomic\_bool \*\_\_a,  
     bool \*\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- template<typename \_ITp >  
     bool `std::atomic_compare_exchange_weak` (\_\_atomic\_base< \_ITp > \*\_\_a,  
     \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool `std::atomic_compare_exchange_weak` (atomic\_address \*\_\_a, void \*\*\_  
     \_\_v1, void \*\_\_v2)
- bool `std::atomic_compare_exchange_weak` (atomic\_bool \*\_\_a, bool \*\_\_i1,  
     bool \_\_i2)
- template<typename \_ITp >  
     bool `std::atomic_compare_exchange_weak_explicit` (\_\_atomic\_base< \_ITp  
     > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `std::atomic_compare_exchange_weak_explicit` (atomic\_bool \*\_\_a, bool  
     \_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `std::atomic_compare_exchange_weak_explicit` (atomic\_address \*\_\_a,  
     void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)



- `void * std::atomic_exchange (atomic_address *__a, void *__v)`
- `template<typename _ITp >  
_ITp std::atomic_exchange (__atomic_base<_ITp> *__a, _ITp __i)`
- `bool std::atomic_exchange (atomic_bool *__a, bool __i)`
- `template<typename _ITp >  
_ITp std::atomic_exchange_explicit (__atomic_base<_ITp> *__a, _ITp __i,  
memory_order __m)`
- `void * std::atomic_exchange_explicit (atomic_address *__a, void *__v,  
memory_order __m)`
- `bool std::atomic_exchange_explicit (atomic_bool *__a, bool __i, memory_  
order __m)`
- `void * std::atomic_fetch_add (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_add (__atomic_base<_ITp> *__a, _ITp __i)`
- `void * std::atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,  
memory_order __m)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_add_explicit (__atomic_base<_ITp> *__a, _ITp __i,  
memory_order __m)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_and (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_and_explicit (__atomic_base<_ITp> *__a, _ITp __i,  
memory_order __m)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_or (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_or_explicit (__atomic_base<_ITp> *__a, _ITp __i,  
memory_order __m)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_sub (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_sub_explicit (__atomic_base<_ITp> *__a, _ITp __i,  
memory_order __m)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,  
memory_order __m)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_xor (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >  
_ITp std::atomic_fetch_xor_explicit (__atomic_base<_ITp> *__a, _ITp __i,  
memory_order __m)`
- `void std::atomic_flag_clear (__atomic_flag_base *__a)`

- void **std::atomic\_flag\_clear\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) \_GLIBCXX\_NOTHROW
- void **std::atomic\_flag\_clear\_explicit** (atomic\_flag \*\_\_a, memory\_order \_\_m)
- bool **std::atomic\_flag\_test\_and\_set** (\_\_atomic\_flag\_base \*\_\_a)
- bool **std::atomic\_flag\_test\_and\_set\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) \_GLIBCXX\_NOTHROW
- bool **std::atomic\_flag\_test\_and\_set\_explicit** (atomic\_flag \*\_\_a, memory\_order \_\_m)
- bool **std::atomic\_is\_lock\_free** (const atomic\_bool \*\_\_a)
- bool **std::atomic\_is\_lock\_free** (const atomic\_address \*\_\_a)
- template<typename \_ITp >  
bool **std::atomic\_is\_lock\_free** (const \_\_atomic\_base<\_ITp> \*\_\_a)
- void \* **std::atomic\_load** (const atomic\_address \*\_\_a)
- template<typename \_ITp >  
\_ITp **std::atomic\_load** (const \_\_atomic\_base<\_ITp> \*\_\_a)
- bool **std::atomic\_load** (const atomic\_bool \*\_\_a)
- bool **std::atomic\_load\_explicit** (const atomic\_bool \*\_\_a, memory\_order \_\_m)
- void \* **std::atomic\_load\_explicit** (const atomic\_address \*\_\_a, memory\_order \_\_m)
- template<typename \_ITp >  
\_ITp **std::atomic\_load\_explicit** (const \_\_atomic\_base<\_ITp> \*\_\_a, memory\_order \_\_m)
- void **std::atomic\_store** (atomic\_bool \*\_\_a, bool \_\_i)
- void **std::atomic\_store** (atomic\_address \*\_\_a, void \*\_\_v)
- template<typename \_ITp >  
void **std::atomic\_store** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)
- void **std::atomic\_store\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- void **std::atomic\_store\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- bool **std::atomic<\_Tp \* >::compare\_exchange\_strong** (\_Tp \* &, \_Tp \*, memory\_order=memory\_order\_seq\_cst)
- bool **std::atomic<\_Tp \* >::compare\_exchange\_strong** (\_Tp \* &, \_Tp \*, memory\_order, memory\_order)
- bool **std::atomic<\_Tp \* >::compare\_exchange\_weak** (\_Tp \* &, \_Tp \*, memory\_order=memory\_order\_seq\_cst)
- bool **std::atomic<\_Tp \* >::compare\_exchange\_weak** (\_Tp \* &, \_Tp \*, memory\_order, memory\_order)
- \_Tp \* **std::atomic<\_Tp \* >::exchange** (\_Tp \*, memory\_order=memory\_order\_seq\_cst)

- `_Tp * std::atomic<_Tp * >::fetch_add` (ptrdiff\_t, memory\_order=memory\_order\_seq\_cst)
- `_Tp * std::atomic<_Tp * >::fetch_sub` (ptrdiff\_t, memory\_order=memory\_order\_seq\_cst)
- `template<typename _Tp > _Tp std::kill_dependency` (\_Tp \_\_y)
- `_Tp * std::atomic<_Tp * >::load` (memory\_order=memory\_order\_seq\_cst) const

### 2.34.1 Detailed Description

Components for performing atomic operations.

### 2.34.2 Define Documentation

#### 2.34.2.1 `#define _GLIBCXX_ATOMIC_PROPERTY`

29.2 Lock-free Property

Definition at line 68 of file `atomic_base.h`.

### 2.34.3 Typedef Documentation

#### 2.34.3.1 `typedef __atomic_base<char> atomic_char`

`atomic_char`

Definition at line 71 of file `atomicfwd_cxx.h`.

#### 2.34.3.2 `typedef __atomic_base<char16_t> atomic_char16_t`

`atomic_char16_t`

Definition at line 107 of file `atomicfwd_cxx.h`.

#### 2.34.3.3 `typedef __atomic_base<char32_t> atomic_char32_t`

`atomic_char32_t`

Definition at line 110 of file `atomicfwd_cxx.h`.

**2.34.3.4 typedef \_\_atomic\_base<int> atomic\_int**

atomic\_int

Definition at line 86 of file atomicfwd\_cxx.h.

**2.34.3.5 typedef \_\_atomic\_base<long long> atomic\_llong**

atomic\_llong

Definition at line 98 of file atomicfwd\_cxx.h.

**2.34.3.6 typedef \_\_atomic\_base<long> atomic\_long**

atomic\_long

Definition at line 92 of file atomicfwd\_cxx.h.

**2.34.3.7 typedef \_\_atomic\_base<signed char> atomic\_schar**

atomic\_schar

Definition at line 74 of file atomicfwd\_cxx.h.

**2.34.3.8 typedef \_\_atomic\_base<short> atomic\_short**

atomic\_short

Definition at line 80 of file atomicfwd\_cxx.h.

**2.34.3.9 typedef \_\_atomic\_base<unsigned char> atomic\_uchar**

atomic\_uchar

Definition at line 77 of file atomicfwd\_cxx.h.

**2.34.3.10 typedef \_\_atomic\_base<unsigned int> atomic\_uint**

atomic\_uint

Definition at line 89 of file atomicfwd\_cxx.h.

**2.34.3.11 typedef \_\_atomic\_base<unsigned long long> atomic\_ullong**

atomic\_ullong

Definition at line 101 of file atomicfwd\_cxx.h.

**2.34.3.12 typedef \_\_atomic\_base<unsigned long> atomic\_ulong**

atomic\_ulong

Definition at line 95 of file atomicfwd\_cxx.h.

**2.34.3.13 typedef \_\_atomic\_base<unsigned short> atomic\_ushort**

atomic\_ushort

Definition at line 83 of file atomicfwd\_cxx.h.

**2.34.3.14 typedef \_\_atomic\_base<wchar\_t> atomic\_wchar\_t**

atomic\_wchar\_t

Definition at line 104 of file atomicfwd\_cxx.h.

**2.34.3.15 typedef enum std::memory\_order std::memory\_order**

Enumeration for memory\_order.

**2.34.4 Enumeration Type Documentation****2.34.4.1 enum std::memory\_order**

Enumeration for memory\_order.

Definition at line 47 of file atomic\_base.h.

**2.34.5 Function Documentation****2.34.5.1 template<typename \_Tp > \_Tp std::kill\_dependency ( \_Tp \_\_y )  
[inline]**

kill\_dependency

Definition at line 54 of file atomic.

## 2.35 Hashes

Collaboration diagram for Hashes:



### Classes

- struct `std::hash<_Tp>`  
*Primary class template hash.*
- struct `std::hash<_Tp*>`  
*Partial specializations for pointer types.*

### Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

### 2.35.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

## 2.36 Locales

### Classes

- class `std::codecvt<_InternT, _ExternT, _StateT>`  
*Primary class template codecvt.*  
*NB: Generic, mostly useless implementation.*
- class `std::ctype<_CharT>`

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets. It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

- class `std::ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

- class `std::ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

- class `std::locale`

Container class for localization functionality.

The `locale` class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

- class `std::locale::facet`

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class `std::locale::id`

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member; otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

- class `std::messages< _CharT >`

Primary class template `messages`.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

- struct `std::messages_base`

Messages facet base class providing catalog typedef.

- class `std::money_base`

*Money format ordering data.*

*This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*

- class `std::money_get<_CharT, _InIter>`

*Primary class template `money_get`.*

*This facet encapsulates the code to parse and return a monetary amount from a string.*

- class `std::money_put<_CharT, _OutIter>`

*Primary class template `money_put`.*

*This facet encapsulates the code to format and output a monetary amount.*

- class `std::moneypunct<_CharT, _Intl>`

*Primary class template `moneypunct`.*

*This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*

- class `std::num_get<_CharT, _InIter>`

*Primary class template `num_get`.*

*This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*

- class `std::num_put<_CharT, _OutIter>`

*Primary class template `num_put`.*

*This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*

- class `std::numpunct<_CharT>`

*Primary class template `numpunct`.*

*This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*

- class `std::time_base`

*Time format ordering data.*

*This class provides an enum representing different orderings of time: day, month, and year.*

- class `std::time_get<_CharT, _InIter>`

*Primary class template `time_get`.*

*This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*



- class `std::time_put<_CharT, _OutIter >`

*Primary class template `time_put`.*

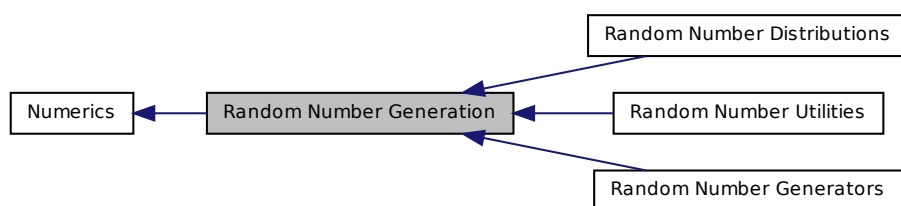
*This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*

### 2.36.1 Detailed Description

Classes and functions for internationalization and localization.

## 2.37 Random Number Generation

Collaboration diagram for Random Number Generation:



### Namespaces

- namespace `std::__detail`

### Modules

- [Random Number Generators](#)
- [Random Number Distributions](#)
- [Random Number Utilities](#)

### Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`  
`_RealType std::generate\_canonical (_UniformRandomNumberGenerator &__g)`

### 2.37.1 Detailed Description

A facility for generating random numbers on selected distributions.

### 2.37.2 Function Documentation

**2.37.2.1** `template<typename _RealType, size_t __bits, typename  
_UniformRandomNumberGenerator > _RealType  
std::generate_canonical ( _UniformRandomNumberGenerator & __g  
)`

A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 2799 of file random.tcc.

References `std::log()`, and `std::min()`.

## 2.38 Regular Expressions

### Classes

- class `std::basic_regex< _Ch_type, _Rx_traits >`
- class `std::match_results< _Bi_iter, _Allocator >`  
*The results of a match or search operation.*
- class `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- struct `std::regex_traits< _Ch_type >`  
*Describes aspects of a regular expression.*
- class `std::sub_match< _BiIter >`

### Typedefs

- typedef `match_results< const char * >` **`std::cmatch`**
- typedef `regex_iterator< const char * >` **`std::cregex_iterator`**
- typedef `regex_token_iterator< const char * >` **`std::cregex_token_iterator`**
- typedef `sub_match< const char * >` **`std::csub_match`**
- typedef `basic_regex< char >` **`std::regex`**
- typedef `match_results< string::const_iterator >` **`std::smatch`**
- typedef `regex_iterator< string::const_iterator >` **`std::sregex_iterator`**

- typedef regex\_token\_iterator< string::const\_iterator > [std::sregex\\_token\\_iterator](#)
- typedef sub\_match< string::const\_iterator > [std::ssub\\_match](#)
- typedef match\_results< const wchar\_t \* > **std::wcmatch**
- typedef regex\_iterator< const wchar\_t \* > **std::wcregex\_iterator**
- typedef regex\_token\_iterator< const wchar\_t \* > [std::wcregex\\_token\\_iterator](#)
- typedef sub\_match< const wchar\_t \* > [std::wcs\\_sub\\_match](#)
- typedef basic\_regex< wchar\_t > [std::wregex](#)
- typedef match\_results< wstring::const\_iterator > **std::wsmatch**
- typedef regex\_iterator< wstring::const\_iterator > **std::wsregex\_iterator**
- typedef regex\_token\_iterator< wstring::const\_iterator > [std::wsregex\\_token\\_iterator](#)
- typedef sub\_match< wstring::const\_iterator > [std::wssub\\_match](#)

## Functions

- template<typename \_Bi\_iter >  
const sub\_match< \_Bi\_iter > & **std::\_\_unmatched\_sub** ()
- bool [std::regex\\_traits::isctype](#) (\_Ch\_type \_\_c, char\_class\_type \_\_f) const
- template<typename \_BiIter >  
bool [std::operator!=](#) (const sub\_match< \_BiIter > &\_\_lhs, const sub\_match< \_BiIter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool [std::operator!=](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool [std::operator!=](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc >  
bool [std::operator!=](#) (const basic\_string< typename iterator\_traits< \_Bi\_iter >::value\_type, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool [std::operator!=](#) (typename iterator\_traits< \_Bi\_iter >::value\_type const \* \_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter, class \_Allocator >  
bool [std::operator!=](#) (const match\_results< \_Bi\_iter, \_Allocator > &\_\_m1, const match\_results< \_Bi\_iter, \_Allocator > &\_\_m2)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc >  
bool [std::operator!=](#) (const sub\_match< \_Bi\_iter > &\_\_lhs, const basic\_string< typename iterator\_traits< \_Bi\_iter >::value\_type, \_Ch\_traits, \_Ch\_alloc > &\_\_rhs)

- `template<typename _Bi_iter >`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_ -`  
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _BiIter >`  
`bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match< _`  
`_BiIter > &__rhs)`
- `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const basic_string<`  
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_`  
`__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >`  
`bool std::operator< (const basic_string< typename iterator_traits< _Bi_iter`  
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_ -`  
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const * -`  
`__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_ -`  
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter >`  
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream<`  
`_Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const basic_ -`  
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`  
`alloc > &__rhs)`
- `template<typename _BiIter >`  
`bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match<`  
`_BiIter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`  
`*__lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator<= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`  
`bool std::operator== (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _BiIter >`  
`bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator> (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_-`  
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *_-`  
`__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`  
`bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match<`  
`_BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const basic_-`  
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_-`  
`alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator>= (const basic_string< typename iterator_traits< _Bi_iter`  
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`  
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`  
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match<`  
`_BiIter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`  
`void std::swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results<`  
`_Bi_iter, _Allocator > &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _-`  
`Ch_type, _Rx_traits > &__rhs)`
- `int std::regex_traits::value (_Ch_type __ch, int __radix) const`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _`  
`Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`  
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_`  
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`  
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type`  
`*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type,`  
`typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc`  
`> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`  
`Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`  
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`  
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_`  
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`  
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_`  
`traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_`  
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`  
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_`  
`iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_`  
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Allocator, class _Rx_traits >`  
`bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *`  
`, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_`  
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_`

```

type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _
Rx_traits >
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_
allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_
constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type,
typename _Rx_traits >
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_
alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_
traits, _Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex<
_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
• template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >
_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __
last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string<
_Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_
constants::match_default)
• template<typename _Rx_traits, typename _Ch_type >
basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_
type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const
basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __
flags=regex_constants::match_default)

```

### 2.38.1 Detailed Description

A facility for performing regular expression pattern matching.

### 2.38.2 Typedef Documentation

#### 2.38.2.1 `typedef regex_token_iterator<const char*> std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

Definition at line 2415 of file regex.h.

#### 2.38.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 847 of file regex.h.



**2.38.2.3    typedef basic\_regex<char> std::regex**

Standard regular expressions.

Definition at line 722 of file regex.h.

**2.38.2.4    typedef regex\_token\_iterator<string::const\_iterator>  
             std::sregex\_token\_iterator**

Token iterator for standard strings.

Definition at line 2417 of file regex.h.

**2.38.2.5    typedef sub\_match<string::const\_iterator> std::ssub\_match**

Standard regex submatch over a standard string.

Definition at line 849 of file regex.h.

**2.38.2.6    typedef regex\_token\_iterator<const wchar\_t\*>  
             std::wcregex\_token\_iterator**

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2420 of file regex.h.

**2.38.2.7    typedef sub\_match<const wchar\_t\*> std::wcsub\_match**

Regex submatch over a C-style null-terminated wide string.

Definition at line 852 of file regex.h.

**2.38.2.8    typedef basic\_regex<wchar\_t> std::wregex**

Standard wide-character regular expressions.

Definition at line 725 of file regex.h.

**2.38.2.9    typedef regex\_token\_iterator<wstring::const\_iterator>  
             std::wsregex\_token\_iterator**

Token iterator for standard wide-character strings.

Definition at line 2422 of file regex.h.

### 2.38.2.10 `typedef sub_match<wstring::const_iterator> std::wssub_match`

Regex submatch over a standard wide string.

Definition at line 854 of file regex.h.

## 2.38.3 Function Documentation

### 2.38.3.1 `template<typename _Ch_type > bool std::regex_traits<_Ch_type>::isctype ( _Ch_type __c, char_class_type __f ) const [inherited]`

Determines if `c` is a member of an identified class.

#### Parameters

`c` a character.

`f` a class type (as returned from `lookup_classname`).

#### Returns

true if the character `c` is a member of the classification represented by `f`, false otherwise.

#### Exceptions

[\*`std::bad\_cast`\*](#) if the current locale does not have a ctype facet.

Definition at line 283 of file regex.h.

References `std::__ctype_abstract_base<_CharT >::is()`, `std::regex_traits<_Ch_type>::lookup_classname()`, `std::use_facet()`, and `std::__ctype_abstract_base<_CharT>::widen()`.

### 2.38.3.2 `template<typename _BiIter > bool std::operator!= ( const sub_match<_BiIter > & __lhs, const sub_match<_BiIter > & __rhs ) [inline]`

Tests the inequivalence of two regular expression submatches.

#### Parameters

`lhs` First regular expression submatch.

`rhs` Second regular expression submatch.

#### Returns

true if `lhs` is not equivalent to `rhs`, false otherwise.

Definition at line 879 of file regex.h.

References `std::sub_match<_BiIter>::compare()`.

**2.38.3.3** `template<typename _Bi_iter> bool std::operator!=( typename  
iterator_traits<_Bi_iter>::value_type const & __lhs, const  
sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

#### Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1268 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.4** `template<typename _Bi_iter> bool std::operator!=( const  
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>  
::value_type const & __rhs ) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A const string reference.

#### Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1342 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.5** `template<typename _Bi_iter, typename _Ch_traits, typename  
_Ch_alloc> bool std::operator!=( const basic_string< typename  
iterator_traits<_Bi_iter>::value_type, _Ch_traits, _Ch_alloc> &  
__lhs, const sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 955 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.6** `template<typename _Bi_iter > bool std::operator!= ( typename  
iterator_traits< _Bi_iter >::value_type const * __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

**Parameters**

*lhs* A regular expression submatch.

*rhs* A string.

**Returns**

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1120 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.7** `template<typename _Bi_iter, class _Allocator > bool std::operator!=  
( const match_results< _Bi_iter, _Allocator > & __m1, const  
match_results< _Bi_iter, _Allocator > & __m2 ) [inline]`

Compares two [match\\_results](#) for inequality.

**Returns**

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1779 of file regex.h.

**2.38.3.8** `template<typename _Bi_iter > bool std::operator!= ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const * __rhs ) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A pointer to a string.

#### Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1194 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.9** `template<typename _Bi_iter , typename _Ch_traits , typename  
_Ch_alloc > bool std::operator!= ( const sub_match< _Bi_iter >  
& __lhs, const basic_string< typename iterator_traits< _Bi_iter  
>::value_type, _Ch_traits, _Ch_alloc > & __rhs ) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

#### Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1036 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.10** `template<typename _BiIter > bool std::operator< ( const  
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &  
__rhs ) [inline]`

Tests the ordering of two regular expression submatches.

#### Parameters

*lhs* First regular expression submatch.

*rhs* Second regular expression submatch.

### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 891 of file regex.h.

```
2.38.3.11 template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
 bool std::operator< (const sub_match< _Bi_iter > & __lhs, const
 basic_string< typename iterator_traits< _Bi_iter >::value_type,
 _Ch_traits, _Ch_alloc > & __rhs) [inline]
```

Tests the ordering of a regular expression submatch and a string.

### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1050 of file regex.h.

```
2.38.3.12 template<typename _Bi_iter > bool std::operator< (typename
 iterator_traits< _Bi_iter >::value_type const & __lhs, const
 sub_match< _Bi_iter > & __rhs) [inline]
```

Tests the ordering of a string and a regular expression submatch.

### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1280 of file regex.h.

**2.38.3.13** `template<typename _Bi_iter, typename _Ch_traits, typename  
_Ch_alloc > bool std::operator< ( const basic_string< typename  
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &  
__lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

#### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 968 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.14** `template<typename _Bi_iter > bool std::operator< ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A const string reference.

#### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1354 of file regex.h.

**2.38.3.15** `template<typename _Bi_iter > bool std::operator< ( typename  
iterator_traits< _Bi_iter >::value_type const * __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1132 of file regex.h.

**2.38.3.16** `template<typename _Bi_iter > bool std::operator< ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

### Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1206 of file regex.h.

**2.38.3.17** `template<typename _Ch_type, typename _Ch_traits, typename  
_Bi_iter > basic_ostream< _Ch_type, _Ch_traits> & std::operator<<  
( basic_ostream< _Ch_type, _Ch_traits > & __os, const  
sub_match< _Bi_iter > & __m ) [inline]`

Inserts a matched string into an output stream.

### Parameters

*os* The output stream.

*m* A submatch string.

### Returns

the output stream with the submatch string inserted.

Definition at line 1405 of file regex.h.



**2.38.3.18** `template<typename _Bi_iter > bool std::operator<= ( typename  
iterator_traits< _Bi_iter >::value_type const & __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

#### Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1316 of file regex.h.

**2.38.3.19** `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >  
bool std::operator<= ( const sub_match< _Bi_iter > & __lhs, const  
basic_string< typename iterator_traits< _Bi_iter >::value_type,  
_Ch_traits, _Ch_alloc > & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

#### Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1092 of file regex.h.

**2.38.3.20** `template<typename _BiIter > bool std::operator<= ( const  
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &  
__rhs ) [inline]`

Tests the ordering of two regular expression submatches.

#### Parameters

*lhs* First regular expression submatch.

*rhs* Second regular expression submatch.

**Returns**

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 903 of file regex.h.

**2.38.3.21** `template<typename _Bi_iter > bool std::operator<= ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

*lhs* A regular expression submatch.

*rhs* A const string reference.

**Returns**

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1390 of file regex.h.

**2.38.3.22** `template<typename _Bi_iter > bool std::operator<= ( typename  
iterator_traits< _Bi_iter >::value_type const * __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1168 of file regex.h.

**2.38.3.23** `template<typename _Bi_iter , typename _Ch_traits , typename  
_Ch_alloc > bool std::operator<= ( const basic_string< typename  
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &  
__lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1007 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.24** `template<typename _Bi_iter > bool std::operator<= ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

*lhs* A regular expression submatch.

*rhs* A string.

**Returns**

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1242 of file regex.h.

**2.38.3.25** `template<typename _Bi_iter > bool std::operator== ( typename  
iterator_traits< _Bi_iter >::value_type const & __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the equivalence of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1255 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.26** `template<typename _Bi_iter , typename _Ch_traits , typename  
_Ch_alloc > bool std::operator==( const sub_match< _Bi_iter >  
& __lhs, const basic_string< typename iterator_traits< _Bi_iter  
>::value_type, _Ch_traits, _Ch_alloc > & __rhs ) [inline]`

Tests the equivalence of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

#### Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1021 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.27** `template<typename _Bi_iter > bool std::operator==( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const & __rhs ) [inline]`

Tests the equivalence of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A const string reference.

#### Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1329 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.28** `template<typename _Bi_iter , typename _Ch_traits , typename  
_Ch_alloc > bool std::operator==( const basic_string< typename  
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &  
__lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the equivalence of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 940 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.29** `template<typename _Bi_iter > bool std::operator==( typename  
iterator_traits< _Bi_iter >::value_type const * __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the equivalence of a C string and a regular expression submatch.

**Parameters**

*lhs* A C string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1107 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.30** `template<typename _Bi_iter, typename _Allocator > bool  
std::operator==( const match_results< _Bi_iter, _Allocator > &  
__m1, const match_results< _Bi_iter, _Allocator > & __m2 )  
[inline]`

Compares two [match\\_results](#) for equality.

**Returns**

true if the two objects refer to the same match, false otherwise.

**Todo**

Implement this function.

**2.38.3.31** `template<typename _BiIter > bool std::operator==( const  
sub_match<_BiIter > & __lhs, const sub_match<_BiIter > &  
__rhs ) [inline]`

Tests the equivalence of two regular expression submatches.

#### Parameters

*lhs* First regular expression submatch.

*rhs* Second regular expression submatch.

#### Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 867 of file regex.h.

References `std::sub_match<_BiIter >::compare()`.

**2.38.3.32** `template<typename _Bi_iter > bool std::operator==( const  
sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter  
>::value_type const * __rhs ) [inline]`

Tests the equivalence of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A pointer to a string?

#### Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1181 of file regex.h.

References `std::sub_match<_BiIter >::str()`.

**2.38.3.33** `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >  
bool std::operator>( const sub_match<_Bi_iter > & __lhs, const  
basic_string< typename iterator_traits<_Bi_iter >::value_type,  
_Ch_traits, _Ch_alloc > & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

**Returns**

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1064 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.34** `template<typename _Bi_iter > bool std::operator> ( typename  
iterator_traits<_Bi_iter >::value_type const & __lhs, const  
sub_match<_Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1292 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.35** `template<typename _Bi_iter > bool std::operator> ( const  
sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter  
>::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

*lhs* A regular expression submatch.

*rhs* A const string reference.

**Returns**

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1366 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.36** `template<typename _Bi_iter , typename _Ch_traits , typename  
_Ch_alloc > bool std::operator> ( const basic_string< typename  
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &  
__lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

#### Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 981 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.37** `template<typename _Bi_iter > bool std::operator> ( typename  
iterator_traits< _Bi_iter >::value_type const * __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

#### Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1144 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.38** `template<typename _BiIter > bool std::operator> ( const  
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &  
__rhs ) [inline]`

Tests the ordering of two regular expression submatches.

#### Parameters

*lhs* First regular expression submatch.



*rhs* Second regular expression submatch.

**Returns**

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 927 of file regex.h.

References `std::sub_match<_BiIter>::compare()`.

**2.38.3.39** `template<typename _Bi_iter > bool std::operator> ( const  
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter  
>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

**Parameters**

*lhs* A regular expression submatch.

*rhs* A string.

**Returns**

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1218 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.40** `template<typename _Bi_iter > bool std::operator>= ( typename  
iterator_traits<_Bi_iter>::value_type const & __lhs, const  
sub_match<_Bi_iter> & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1304 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

**2.38.3.41** `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >  
bool std::operator>= ( const sub_match< _Bi_iter > & __lhs, const  
basic_string< typename iterator_traits< _Bi_iter >::value_type,  
_Ch_traits, _Ch_alloc > & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

#### Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1078 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.42** `template<typename _Bi_iter > bool std::operator>= ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const & __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A const string reference.

#### Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1378 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.43** `template<typename _Bi_iter > bool std::operator>= ( typename  
iterator_traits< _Bi_iter >::value_type const * __lhs, const  
sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

#### Parameters

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1156 of file regex.h.

References `std::sub_match<_Biter>::str()`.

**2.38.3.44** `template<typename _Bi_iter , typename _Ch_traits , typename  
_Ch_alloc > bool std::operator>= ( const basic_string< typename  
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &  
__lhs, const sub_match< _Bi_iter > & __rhs ) [inline]`

Tests the ordering of a string and a regular expression submatch.

**Parameters**

*lhs* A string.

*rhs* A regular expression submatch.

**Returns**

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 994 of file regex.h.

References `std::sub_match<_Biter>::str()`.

**2.38.3.45** `template<typename _Biter > bool std::operator>= ( const  
sub_match< _Biter > & __lhs, const sub_match< _Biter > &  
__rhs ) [inline]`

Tests the ordering of two regular expression submatches.

**Parameters**

*lhs* First regular expression submatch.

*rhs* Second regular expression submatch.

**Returns**

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 915 of file regex.h.

References `std::sub_match<_Biter>::compare()`.

**2.38.3.46** `template<typename _Bi_iter > bool std::operator>= ( const  
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
>::value_type const * __rhs ) [inline]`

Tests the ordering of a regular expression submatch and a string.

#### Parameters

*lhs* A regular expression submatch.

*rhs* A string.

#### Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1230 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

**2.38.3.47** `template<typename _Ch_traits , typename _Ch_alloc , typename  
_Allocator , typename _Ch_type , typename _Rx_traits >  
bool std::regex_match ( const basic_string< _Ch_type,  
_Ch_traits, _Ch_alloc > & __s, match_results< typename  
basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,  
_Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits  
> & __re, regex_constants::match_flag_type __flags =  
regex_constants::match_default ) [inline]`

Determines if there is a match between the regular expression *e* and a string.

#### Parameters

*s* The string to match.

*m* The match results.

*re* The regular expression.

*flags* Controls how the regular expression is matched.

#### Return values

*true* A match exists.

*false* Otherwise.

#### Exceptions

*an* exception of type `regex_error`.

Definition at line 1903 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_match()`.

```
2.38.3.48 template<typename _Ch_type , typename _Allocator ,
 typename _Rx_traits > bool std::regex_match (const
 _Ch_type * __s, match_results< const _Ch_type *, _Allocator
 > & __m, const basic_regex< _Ch_type, _Rx_traits
 > & __re, regex_constants::match_flag_type __f =
 regex_constants::match_default) [inline]
```

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

#### Parameters

- s* The C-style null-terminated string to match.
- m* The match results.
- re* The regular expression.
- f* Controls how the regular expression is matched.

#### Return values

- true* A match exists.
- false* Otherwise.

#### Exceptions

- an* exception of type `regex_error`.

Definition at line 1879 of file `regex.h`.

References `std::regex_match()`.

```
2.38.3.49 template<typename _Ch_type , class _Rx_traits > bool
 std::regex_match (const _Ch_type * __s, const basic_regex<
 _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type
 __f = regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

#### Parameters

- s* The C-style null-terminated string to match.

*re* The regular expression.

*f* Controls how the regular expression is matched.

### Return values

*true* A match exists.

*false* Otherwise.

### Exceptions

*an* exception of type [regex\\_error](#).

Definition at line 1926 of file `regex.h`.

References `std::regex_match()`.

```
2.38.3.50 template<typename _Ch_traits , typename _Str_allocator ,
typename _Ch_type , typename _Rx_traits > bool std::regex_match
(const basic_string< _Ch_type, _Ch_traits, _Str_allocator
> & __s, const basic_regex< _Ch_type, _Rx_traits >
& __re, regex_constants::match_flag_type __flags =
regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression `e` and a string.

### Parameters

*s* [IN] The string to match.

*re* [IN] The regular expression.

*flags* [IN] Controls how the regular expression is matched.

### Return values

*true* A match exists.

*false* Otherwise.

### Exceptions

*an* exception of type [regex\\_error](#).

Definition at line 1948 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```

2.38.3.51 template<typename _Bi_iter , typename _Allocator , typename
 _Ch_type , typename _Rx_traits > bool std::regex_match
 (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter,
 _Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits
 > & __re, regex_constants::match_flag_type __flags =
 regex_constants::match_default)

```

Determines if there is a match between the regular expression *e* and all of the character sequence [*first*, *last*).

#### Parameters

- s* Start of the character sequence to match.
- e* One-past-the-end of the character sequence to match.
- m* The match results.
- re* The regular expression.
- flags* Controls how the regular expression is matched.

#### Return values

- true* A match exists.
- false* Otherwise.

#### Exceptions

- an* exception of type `regex_error`.

#### Todo

- Implement this function.

Definition at line 1823 of file `regex.h`.

Referenced by `std::regex_match()`.

```

2.38.3.52 template<typename _Bi_iter , typename _Ch_type , typename
 _Rx_traits > bool std::regex_match (_Bi_iter __first,
 _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
 > & __re, regex_constants::match_flag_type __flags =
 regex_constants::match_default)

```

Indicates if there is a match between the regular expression *e* and all of the character sequence [*first*, *last*).

#### Parameters

- first* Beginning of the character sequence to match.

*last* One-past-the-end of the character sequence to match.

*re* The regular expression.

*flags* Controls how the regular expression is matched.

### Return values

*true* A match exists.

*false* Otherwise.

### Exceptions

*an* exception of type [regex\\_error](#).

Definition at line 1854 of file `regex.h`.

References `std::regex_match()`.

```
2.38.3.53 template<typename _Rx_traits , typename _Ch_type
> basic_string<_Ch_type> std::regex_replace (const
basic_string<_Ch_type> & __s, const basic_regex<
_Ch_type, _Rx_traits> & __e, const basic_string<_Ch_type
> & __fmt, regex_constants::match_flag_type __flags =
regex_constants::match_default) [inline]
```

### Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

### Parameters

*s*

*e*

*fmt*

*flags*

### Returns

a copy of string *s* with replacements.

### Exceptions

*an* exception of type [regex\\_error](#).

Definition at line 2127 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_replace()`.



```

2.38.3.54 template<typename _Out_iter , typename _Bi_iter , typename
 _Rx_traits , typename _Ch_type > _Out_iter std::regex_replace
 (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const
 basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string<
 _Ch_type > & __fmt, regex_constants::match_flag_type __flags =
 regex_constants::match_default) [inline]

```

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

#### Parameters

*out*  
*first*  
*last*  
*e*  
*fmt*  
*flags*

#### Returns

out

#### Exceptions

*an* exception of type [regex\\_error](#).

#### Todo

Implement this function.

Definition at line 2107 of file regex.h.

Referenced by `std::regex_replace()`.

```

2.38.3.55 template<typename _Ch_traits , typename _Ch_alloc , typename
 _Allocator , typename _Ch_type , typename _Rx_traits >
 bool std::regex_search (const basic_string< _Ch_type,
 _Ch_traits, _Ch_alloc > & __s, match_results< typename
 basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
 _Allocator > & __m, const basic_regex< _Ch_type,
 _Rx_traits > & __e, regex_constants::match_flag_type __f =
 regex_constants::match_default) [inline]

```

Searches for a regular expression within a string.

**Parameters**

*s* [IN] A C++ string to search for the regex.  
*m* [OUT] The set of regex matches.  
*e* [IN] The regex to search for in *s*.  
*f* [IN] The search flags.

**Return values**

*true* A match was found within the string.  
*false* No match was found within the string, the content of *m* is undefined.

**Exceptions**

*an* exception of type [regex\\_error](#).

Definition at line 2081 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

**2.38.3.56** `template<typename _Bi_iter, typename _Allocator, typename  
 _Ch_type, typename _Rx_traits > bool std::regex_search  
 ( _Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter,  
 _Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits  
 > & __re, regex_constants::match_flag_type __flags =  
 regex_constants::match_default ) [inline]`

Searches for a regular expression within a range.

**Parameters**

*first* [IN] The start of the string to search.  
*last* [IN] One-past-the-end of the string to search.  
*m* [OUT] The match results.  
*re* [IN] The regular expression to search for.  
*flags* [IN] Search policy flags.

**Return values**

*true* A match was found within the string.  
*false* No match was found within the string, the content of *m* is undefined.

**Exceptions**

*an* exception of type [regex\\_error](#).

**Todo**

Implement this function.

Definition at line 1973 of file regex.h.

Referenced by std::regex\_search().

```
2.38.3.57 template<typename _Ch_type , typename _Rx_traits > bool
 std::regex_search (const _Ch_type * __s, const basic_regex<
 _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type
 __f= regex_constants::match_default) [inline]
```

Searches for a regular expression within a C-string.

**Parameters**

*s* [IN] The C-string to search.

*e* [IN] The regular expression to search for.

*f* [IN] Search policy flags.

**Return values**

*true* A match was found within the string.

*false* No match was found within the string.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Exceptions**

*an* exception of type [regex\\_error](#).

Definition at line 2038 of file regex.h.

References std::regex\_search().

```
2.38.3.58 template<typename _Ch_type , class _Allocator , class _Rx_traits >
 bool std::regex_search (const _Ch_type * __s, match_results< const
 _Ch_type *, _Allocator > & __m, const basic_regex< _Ch_type,
 _Rx_traits > & __e, regex_constants::match_flag_type __f=
 regex_constants::match_default) [inline]
```

Searches for a regular expression within a C-string.

**Parameters**

*s* [IN] A C-string to search for the regex.

*m* [OUT] The set of regex matches.

*e* [IN] The regex to search for in *s*.

*f* [IN] The search flags.

**Return values**

*true* A match was found within the string.

*false* No match was found within the string, the content of *m* is undefined.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

**Exceptions**

*an* exception of type [regex\\_error](#).

Definition at line 2018 of file `regex.h`.

References `std::regex_search()`.

```
2.38.3.59 template<typename _Bi_iter , typename _Ch_type , typename
 _Rx_traits > bool std::regex_search (_Bi_iter __first,
 _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
 > & __re, regex_constants::match_flag_type __flags =
 regex_constants::match_default) [inline]
```

Searches for a regular expression within a range.

**Parameters**

*first* [IN] The start of the string to search.

*last* [IN] One-past-the-end of the string to search.

*re* [IN] The regular expression to search for.

*flags* [IN] Search policy flags.

**Return values**

*true* A match was found within the string.

*false* No match was found within the string.

### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

### Exceptions

*an* exception of type `regex_error`.

Definition at line 1994 of file `regex.h`.

References `std::regex_search()`.

```
2.38.3.60 template<typename _Ch_traits , typename _String_allocator ,
 typename _Ch_type , typename _Rx_traits > bool std::regex_search
(const basic_string< _Ch_type, _Ch_traits, _String_allocator
 > & __s, const basic_regex< _Ch_type, _Rx_traits >
 & __e, regex_constants::match_flag_type __flags =
 regex_constants::match_default) [inline]
```

Searches for a regular expression within a string.

### Parameters

*s* [IN] The string to search.

*e* [IN] The regular expression to search for.

*flags* [IN] Search policy flags.

### Return values

*true* A match was found within the string.

*false* No match was found within the string.

### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

### Exceptions

*an* exception of type `regex_error`.

Definition at line 2058 of file `regex.h`.

References `std::regex_search()`.

**2.38.3.61** `template<typename _Ch_type, typename _Rx_traits> void  
std::swap ( basic_regex< _Ch_type, _Rx_traits> & __lhs,  
basic_regex< _Ch_type, _Rx_traits> & __rhs ) [inline]`

Swaps the contents of two regular expression objects.

#### Parameters

*lhs* First regular expression.

*rhs* Second regular expression.

Definition at line 737 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits>::swap()`.

**2.38.3.62** `template<typename _Bi_iter, typename _Allocator> void std::swap  
( match_results< _Bi_iter, _Allocator> & __lhs, match_results<  
_Bi_iter, _Allocator> & __rhs ) [inline]`

Swaps two match results.

#### Parameters

*lhs* A match result.

*rhs* A match result.

The contents of the two [match\\_results](#) objects are swapped.

Definition at line 1793 of file regex.h.

References `std::match_results< _Bi_iter, _Allocator>::swap()`.

**2.38.3.63** `template<typename _Ch_type> int std::regex_traits< _Ch_type>  
>::value ( _Ch_type __ch, int __radix ) const [inherited]`

Converts a digit to an int.

#### Parameters

*ch* a character representing a digit.

*radix* the radix if the numeric conversion (limited to 8, 10, or 16).

#### Returns

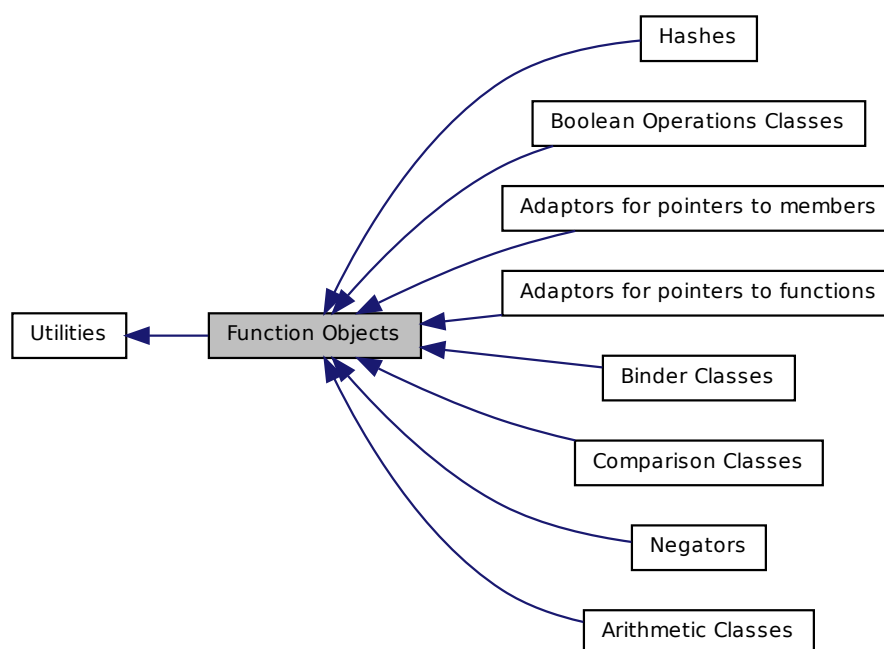
the value represented by the digit *ch* in base *radix* if the character *ch* is a valid digit in base *radix*; otherwise returns -1.

Definition at line 319 of file regex.h.

References `std::basic_ios<_CharT, _Traits>::fail()`.

## 2.39 Function Objects

Collaboration diagram for Function Objects:



### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result>`
- class `std::function<_Res(_ArgTypes...)>`  
*Primary class template for `std::function`.  
 Polymorphic function wrapper.*
- class `std::reference_wrapper<_Tp>`

Primary class template for *reference\_wrapper*.

- struct `std::unary_function<_Arg, _Result>`

## Modules

- [Binder Classes](#)
- [Hashes](#)
- [Arithmetic Classes](#)
- [Comparison Classes](#)
- [Boolean Operations Classes](#)
- [Negators](#)
- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)

## Functions

- template<typename \_Tp, typename \_Class>  
`_Mem_fn<_Tp _Class::*> std::mem_fn(_Tp _Class::* __pm)`

### 2.39.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform(a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.



## 2.39.2 Function Documentation

**2.39.2.1** `template<typename _Tp , typename _Class > _Mem_fn<_Tp  
_Class::*> std::mem_fn ( _Tp _Class::* __pm ) [inline]`

Returns a function object that forwards to the member pointer *pm*.

Definition at line 768 of file functional.

## 2.40 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



### Classes

- struct `std::divides<_Tp>`  
*One of the [math functors](#).*
- struct `std::minus<_Tp>`  
*One of the [math functors](#).*
- struct `std::modulus<_Tp>`  
*One of the [math functors](#).*
- struct `std::multiplies<_Tp>`  
*One of the [math functors](#).*
- struct `std::negate<_Tp>`  
*One of the [math functors](#).*
- struct `std::plus<_Tp>`  
*One of the [math functors](#).*

### 2.40.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

## 2.41 Comparison Classes

Collaboration diagram for Comparison Classes:



### Classes

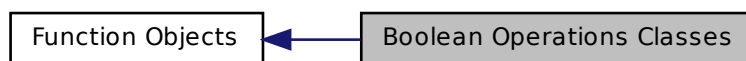
- struct `std::equal_to< _Tp >`  
*One of the [comparison functors](#).*
- struct `std::greater< _Tp >`  
*One of the [comparison functors](#).*
- struct `std::greater_equal< _Tp >`  
*One of the [comparison functors](#).*
- struct `std::less< _Tp >`  
*One of the [comparison functors](#).*
- struct `std::less_equal< _Tp >`  
*One of the [comparison functors](#).*
- struct `std::not_equal_to< _Tp >`  
*One of the [comparison functors](#).*

### 2.41.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like <.

## 2.42 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



### Classes

- struct `std::logical_and< _Tp >`  
*One of the [Boolean operations functors](#).*
- struct `std::logical_not< _Tp >`  
*One of the [Boolean operations functors](#).*
- struct `std::logical_or< _Tp >`  
*One of the [Boolean operations functors](#).*

### 2.42.1 Detailed Description

Here are wrapper functors for Boolean operations: &&, ||, and !.

## 2.43 Negators

Collaboration diagram for Negators:



### Classes

- class `std::binary_negate< _Predicate >`  
*One of the [negation functors](#).*
- class `std::unary_negate< _Predicate >`  
*One of the [negation functors](#).*

### Functions

- `template<typename _Predicate >`  
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`

#### 2.43.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of [unary\\_negate](#) or [binary\\_negate](#), respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
 bool operator() (int x) { return x > 3; }
};

```

```
std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (i) of `v` for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

### 2.43.2 Function Documentation

**2.43.2.1** `template<typename _Predicate> unary_negate<_Predicate>  
std::not1 ( const _Predicate & __pred ) [inline]`

One of the [negation functors](#).

Definition at line 364 of file `stl_function.h`.

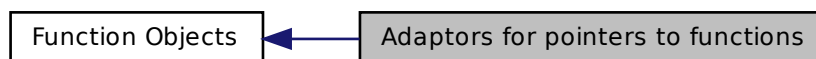
**2.43.2.2** `template<typename _Predicate> binary_negate<_Predicate>  
std::not2 ( const _Predicate & __pred ) [inline]`

One of the [negation functors](#).

Definition at line 389 of file `stl_function.h`.

## 2.44 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



### Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`  
*One of the [adaptors for function pointers](#).*

- class `std::pointer_to_unary_function< _Arg, _Result >`

*One of the [adaptors for function pointers](#).*

## Functions

- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_-`  
`Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_-`  
`Result(*__x)(_Arg1, _Arg2))`

### 2.44.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes [unary\\_function](#) and [binary\\_function](#)). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor [pointer\\_to\\_unary\\_function](#) is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor [pointer\\_to\\_binary\\_function](#) does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

### 2.44.2 Function Documentation

- #### 2.44.2.1
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function<_Arg, _Result> std::ptr_fun (`  
`_Result(*)(_Arg) __x ) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 437 of file `stl_function.h`.

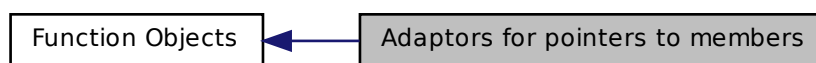
**2.44.2.2** `template<typename _Arg1 , typename _Arg2 , typename _Result >  
 pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (  
 _Result(*)(_Arg1, _Arg2) __x ) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 463 of file `stl_function.h`.

## 2.45 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



### Classes

- class `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun_ref_t< _Ret, _Tp >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun_t< _Ret, _Tp >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun1_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*

- class `std::mem_fun_ref_t<_Ret, _Tp>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun_t<_Ret, _Tp>`  
*One of the [adaptors for member /// pointers](#).*

## Functions

- `template<typename _Ret, typename _Tp>`  
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_-Arg))`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`

### 2.45.1 Detailed Description

There are a total of  $8 = 2^3$  function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

## 2.46 Heap

Collaboration diagram for Heap:





## Functions

- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`

## 2.46.1 Function Documentation

**2.46.1.1** `template<typename _RandomAccessIterator > bool std::is_heap (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`  
`[inline]`

Determines whether a range is a heap.

### Parameters

*first* Start of range.

*last* End of range.

### Returns

True if range is a heap, false otherwise.

Definition at line 558 of file `stl_heap.h`.

References `std::is_heap_until()`.

**2.46.1.2** `template<typename _RandomAccessIterator , typename`  
`_Compare > bool std::is_heap ( _RandomAccessIterator __first,`  
`_RandomAccessIterator __last, _Compare __comp ) [inline]`

Determines whether a range is a heap using comparison functor.

### Parameters

*first* Start of range.

*last* End of range.

*comp* Comparison functor to use.

### Returns

True if range is a heap, false otherwise.

Definition at line 571 of file `stl_heap.h`.

References `std::is_heap_until()`.

**2.46.1.3** `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until ( _RandomAccessIterator`  
`__first, _RandomAccessIterator __last ) [inline]`

Search the end of a heap.

**Parameters***first* Start of range.*last* End of range.**Returns**

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap.

Definition at line 510 of file `stl_heap.h`.

References `std::distance()`.

```
2.46.1.4 template<typename _RandomAccessIterator , typename _Compare >
 _RandomAccessIterator std::is_heap_until (_RandomAccessIterator
 __first, _RandomAccessIterator __last, _Compare __comp)
 [inline]
```

Search the end of a heap using comparison functor.

**Parameters***first* Start of range.*last* End of range.*comp* Comparison functor to use.**Returns**

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap. Comparisons are made using *comp*.

Definition at line 536 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

```
2.46.1.5 template<typename _RandomAccessIterator > void std::make_heap (
 _RandomAccessIterator __first, _RandomAccessIterator __last)
```

Construct a heap over a range.

**Parameters**

*first* Start of heap.

*last* End of heap.

This operation makes the elements in [first,last) into a heap.

Definition at line 373 of file stl\_heap.h.

```
2.46.1.6 template<typename _RandomAccessIterator , typename _Compare
> void std::make_heap (_RandomAccessIterator __first,
 _RandomAccessIterator __last, _Compare __comp)
```

Construct a heap over a range using comparison functor.

**Parameters**

*first* Start of heap.

*last* End of heap.

*comp* Comparison functor to use.

This operation makes the elements in [first,last) into a heap. Comparisons are made using comp.

Definition at line 413 of file stl\_heap.h.

Referenced by std::\_\_heap\_select(), std::partial\_sort\_copy(), and std::priority\_queue<\_Tp, \_Sequence, \_Compare >::priority\_queue().

```
2.46.1.7 template<typename _RandomAccessIterator > void std::pop_heap (
 _RandomAccessIterator __first, _RandomAccessIterator __last)
 [inline]
```

Pop an element off a heap.

**Parameters**

*first* Start of heap.

*last* End of heap.

This operation pops the top of the heap. The elements first and last-1 are swapped and [first,last-1) is made into a heap.

Definition at line 276 of file stl\_heap.h.

**2.46.1.8** `template<typename _RandomAccessIterator , typename _Compare  
> void std::pop_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp ) [inline]`

Pop an element off a heap using comparison functor.

#### Parameters

*first* Start of heap.

*last* End of heap.

*comp* Comparison functor to use.

This operation pops the top of the heap. The elements *first* and *last*-1 are swapped and [*first*,*last*-1) is made into a heap. Comparisons are made using *comp*.

Definition at line 350 of file `stl_heap.h`.

Referenced by `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`.

**2.46.1.9** `template<typename _RandomAccessIterator , typename _Compare  
> void std::push_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp ) [inline]`

Push an element onto a heap using comparison functor.

#### Parameters

*first* Start of heap.

*last* End of heap + element.

*comp* Comparison functor.

This operation pushes the element at *last*-1 onto the valid heap over the range [*first*,*last*-1). After completion, [*first*,*last*) is a valid heap. Compare operations are performed using *comp*.

Definition at line 203 of file `stl_heap.h`.

Referenced by `std::priority_queue< _Tp, _Sequence, _Compare >::push()`.

**2.46.1.10** `template<typename _RandomAccessIterator > void std::push_heap (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`  
`[inline]`

Push an element onto a heap.

**Parameters**

*first* Start of heap.

*last* End of heap + element.

This operation pushes the element at last-1 onto the valid heap over the range [first,last-1). After completion, [first,last) is a valid heap.

Definition at line 154 of file stl\_heap.h.

**2.46.1.11** `template<typename _RandomAccessIterator , typename _Compare  
> void std::sort_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp )`

Sort a heap using comparison functor.

**Parameters**

*first* Start of heap.

*last* End of heap.

*comp* Comparison functor to use.

This operation sorts the valid heap in the range [first,last). Comparisons are made using comp.

Definition at line 481 of file stl\_heap.h.

Referenced by std::partial\_sort(), and std::partial\_sort\_copy().

**2.46.1.12** `template<typename _RandomAccessIterator > void std::sort_heap (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`

Sort a heap.

**Parameters**

*first* Start of heap.

*last* End of heap.

This operation sorts the valid heap in the range [first,last).

Definition at line 452 of file stl\_heap.h.

## 2.47 Iterators

### Classes

- class `std::back_insert_iterator< _Container >`  
*Turns assignment into insertion.*
- struct `std::bidirectional_iterator_tag`  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct `std::forward_iterator_tag`  
*Forward iterators support a superset of input iterator operations.*
- class `std::front_insert_iterator< _Container >`  
*Turns assignment into insertion.*
- struct `std::input_iterator_tag`  
*Marking input iterators.*
- class `std::insert_iterator< _Container >`  
*Turns assignment into insertion.*
- class `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`  
*Provides input iterator semantics for streams.*
- class `std::istreambuf_iterator< _CharT, _Traits >`  
*Provides input iterator semantics for streambufs.*
- struct `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`  
*Common iterator class.*
- struct `std::iterator_traits< _Iterator >`  
*Traits class for iterators.*
- struct `std::iterator_traits< _Tp * >`  
*Partial specialization for pointer types.*
- struct `std::iterator_traits< const _Tp * >`  
*Partial specialization for const pointer types.*
- class `std::move_iterator< _Iterator >`
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`

*Provides output iterator semantics for streams.*

- class [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >](#)  
*Provides output iterator semantics for streambufs.*
- struct [std::output\\_iterator\\_tag](#)  
*Marking output iterators.*
- struct [std::random\\_access\\_iterator\\_tag](#)  
*Random-access iterators support a superset of bidirectional /// iterator operations.*
- class [std::reverse\\_iterator< \\_Iterator >](#)

## Functions

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type` [std::copy\\_move\\_a2](#) (`const _CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result`)
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type` [std::copy\\_move\\_a2](#) (`const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result`)
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type` [std::copy\\_move\\_a2](#) (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result`)
- `template<typename _Iter >`  
`iterator_traits< _Iter >::iterator_category` [std::\\_\\_iterator\\_category](#) (`const _Iter &`)
- `template<typename _Container >`  
`back_insert_iterator< _Container >` [std::back\\_inserter](#) (`_Container &__x`)
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type` [std::copy](#) (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result`)
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type` [std::find](#) (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val`)
- `template<typename _Container >`  
`front_insert_iterator< _Container >` [std::front\\_inserter](#) (`_Container &__x`)



- `template<typename _Container, typename _Iterator>`  
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator>`  
`move_iterator< _Iterator > std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR>`  
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits>`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Iterator>`  
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist>`  
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Iterator>`  
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator>`  
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR>`  
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator>`  
`reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator>`  
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _Iterator >`  
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`

### 2.47.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

### 2.47.2 Function Documentation

#### 2.47.2.1 `template<typename _Iter > iterator_traits<_Iter>::iterator_category std::__iterator_category ( const _Iter & ) [inline]`

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 167 of file `stl_iterator_base_types.h`.

Referenced by `std::advance()`, `std::copy_n()`, `__gnu_cxx::copy_n()`, `std::distance()`, `__gnu_cxx::distance()`, `std::find()`, `std::find_end()`, `std::find_if()`, `std::find_if_not()`, `std::partition()`, `std::reverse()`, `std::search_n()`, `std::uninitialized_copy_n()`, `__gnu_cxx::uninitialized_copy_n()`, and `std::unique_copy()`.

#### 2.47.2.2 `template<typename _Container > back_insert_iterator<_Container> std::back_inserter ( _Container & __x ) [inline]`

##### Parameters

*x* A container of arbitrary type.

##### Returns

An instance of `back_insert_iterator` working on *x*.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 464 of file `stl_iterator.h`.

Referenced by `std::regex_replace()`.

**2.47.2.3** `template<typename _Container > front_insert_iterator<_Container>  
std::front_inserter ( _Container & __x ) [inline]`

#### Parameters

**x** A container of arbitrary type.

#### Returns

An instance of [front\\_insert\\_iterator](#) working on **x**.

This wrapper function helps in creating [front\\_insert\\_iterator](#) instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 547 of file `stl_iterator.h`.

**2.47.2.4** `template<typename _Container , typename _Iterator >  
insert_iterator<_Container> std::inserter ( _Container & __x,  
_Iterator __i ) [inline]`

#### Parameters

**x** A container of arbitrary type.

#### Returns

An instance of [insert\\_iterator](#) working on **x**.

This wrapper function helps in creating [insert\\_iterator](#) instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 653 of file `stl_iterator.h`.

**2.47.2.5** `template<class _Tp , class _CharT , class _Traits , class _Dist > bool  
std::operator!=( const istream_iterator< _Tp, _CharT, _Traits, _Dist  
> & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &  
__y ) [inline]`

Return false if **x** and **y** are both end or not end, or **x** and **y** are the same.

Definition at line 135 of file `stream_iterator.h`.

**2.47.2.6** `template<typename _Iterator > bool std::operator==( const  
reverse_iterator< _Iterator > & __x, const reverse_iterator<  
_Iterator > & __y ) [inline]`

#### Parameters

*x* A reverse\_iterator.

*y* A reverse\_iterator.

#### Returns

A simple bool.

Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 283 of file stl\_iterator.h.

References std::reverse\_iterator< \_Iterator >::base().

**2.47.2.7** `template<typename _Tp , typename _CharT , typename _Traits ,  
typename _Dist > bool std::operator==( const istream_iterator<  
_Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp,  
_CharT, _Traits, _Dist > & __y ) [inline]`

Return true if x and y are both end or not end, or x and y are the same.

Definition at line 128 of file stream\_iterator.h.

## 2.48 Strings

Collaboration diagram for Strings:



## Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`

*Managing sequences of characters and character-like objects.*

## Typedefs

- typedef `basic_string< char >` `std::string`
- typedef `basic_string< char16_t >` `std::u16string`
- typedef `basic_string< char32_t >` `std::u32string`
- typedef `basic_string< wchar_t >` `std::wstring`

### 2.48.1 Typedef Documentation

#### 2.48.1.1 typedef `basic_string<char>` `std::string`

A string of `char`.

Definition at line 63 of file `stringfwd.h`.

#### 2.48.1.2 typedef `basic_string<char16_t>` `std::u16string`

A string of `char16_t`.

Definition at line 77 of file `stringfwd.h`.

#### 2.48.1.3 typedef `basic_string<char32_t>` `std::u32string`

A string of `char32_t`.

Definition at line 78 of file `stringfwd.h`.

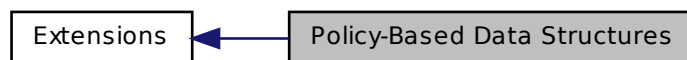
#### 2.48.1.4 typedef `basic_string<wchar_t>` `std::wstring`

A string of `wchar_t`.

Definition at line 68 of file `stringfwd.h`.

## 2.49 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



### Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >`  
*An abstract basic hash-based associative container.*
- class `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`  
*An abstract basic tree-like (tree, trie) associative container.*
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`  
*A concrete collision-chaining hash-based associative container.*
- class `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`  
*An abstract basic associative container.*
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`  
*A concrete general-probing hash-based associative container.*
- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`  
*A list-update based associative container.*
- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`  
*A concrete basic tree-based associative container.*

- class `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`

*A concrete basic trie-based associative container.*

## Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`

### 2.49.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: [http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb\\_ds/index.html](http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html)

## 2.50 Random Number Generators

Collaboration diagram for Random Number Generators:





## Classes

- class `std::discard_block_engine<_RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m >`

*A model of a linear congruential random number generator.*

- class `std::random_device`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k >`

*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.*

## Typedefs

- typedef minstd\_rand0 **std::default\_random\_engine**
- typedef shuffle\_order\_engine< minstd\_rand0, 256 > **std::knuth\_b**
- typedef linear\_congruential\_engine< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > **std::minstd\_rand**
- typedef linear\_congruential\_engine< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > **std::minstd\_rand0**
- typedef mersenne\_twister\_engine< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > **std::mt19937**
- typedef mersenne\_twister\_engine< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > **std::mt19937\_64**
- typedef discard\_block\_engine< ranlux24\_base, 223, 23 > **std::ranlux24**
- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > **std::ranlux24\_base**
- typedef discard\_block\_engine< ranlux48\_base, 389, 11 > **std::ranlux48**
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > **std::ranlux48\_base**

## Functions

- template<typename \_UIntType, \_UIntType \_\_a, \_UIntType \_\_c, \_UIntType \_\_m>  
bool **std::operator!=**(const **std::linear\_congruential\_engine**<\_UIntType, \_\_a, \_\_c, \_\_m > &\_\_lhs, const **std::linear\_congruential\_engine**<\_UIntType, \_\_a, \_\_c, \_\_m > &\_\_rhs)

- `template<typename _RandomNumberEngine, size_t __k>`  
`bool std::operator!= (const std::shuffle_order_engine< _-`  
`RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine<`  
`_RandomNumberEngine, __k > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`  
`UIntType __f>`  
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, _-`  
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`  
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`  
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_-`  
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _-`  
`RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine<`  
`_RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w,`  
`__s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w,`  
`__s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`  
`, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__x)`

### 2.50.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.

Table 2.1: Random Number Generator Requirements

## 2.50.2 Typedef Documentation

### 2.50.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1473 of file random.h.

### 2.50.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

Definition at line 1467 of file random.h.

### 2.50.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1489 of file random.h.

### 2.50.2.4 `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1501 of file random.h.

### 2.50.3 Function Documentation

**2.50.3.1** `template<typename _UIntType , _UIntType __a, _UIntType  
__c, _UIntType __m> bool std::operator!= ( const  
std::linear_congruential_engine< _UIntType, __a, __c, __m > &  
__lhs, const std::linear_congruential_engine< _UIntType, __a, __c,  
__m > & __rhs ) [inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A linear congruential random number generator object.

`__rhs` Another linear congruential random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 331 of file random.h.

**2.50.3.2** `template<typename _RandomNumberEngine , size_t __k>  
bool std::operator!= ( const std::shuffle_order_engine<  
_RandomNumberEngine, __k > & __lhs, const  
std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs  
) [inline]`

Compares two shuffle\_order\_engine random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A shuffle\_order\_engine random number generator object.

`__rhs` Another shuffle\_order\_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1456 of file random.h.

```

2.50.3.3 template<typename _UIntType , size_t __w, size_t __n, size_t __m,
size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
_UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f> bool std::operator!=(const std::mersenne_twister_engine<
_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l,
__f> & __lhs, const std::mersenne_twister_engine< _UIntType, __w,
__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __rhs)
[inline]

```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for inequality.

#### Parameters

**\_\_lhs** A % mersenne\_twister\_engine random number generator object.

**\_\_rhs** Another % mersenne\_twister\_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 567 of file random.h.

```

2.50.3.4 template<typename _RandomNumberEngine , size_t
__w, typename _UIntType > bool std::operator!=(const
std::independent_bits_engine< _RandomNumberEngine, __w,
_UIntType > & __lhs, const std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType > & __rhs) [inline]

```

Compares two independent\_bits\_engine random number generator objects of the same type for inequality.

#### Parameters

**\_\_lhs** A independent\_bits\_engine random number generator object.

**\_\_rhs** Another independent\_bits\_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1200 of file random.h.

**2.50.3.5** `template<typename _RandomNumberEngine, size_t __p, size_t __r> bool std::operator!=( const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs ) [inline]`

Compares two `discard_block_engine` random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A `discard_block_engine` random number generator object.

`__rhs` Another `discard_block_engine` random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 998 of file `random.h`.

**2.50.3.6** `template<typename _UIntType, size_t __w, size_t __s, size_t __r> bool std::operator!=( const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs ) [inline]`

Compares two `% subtract_with_carry_engine` random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A `% subtract_with_carry_engine` random number generator object.

`__rhs` Another `% subtract_with_carry_engine` random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 770 of file `random.h`.

```

2.50.3.7 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType, typename _CharT, typename _Traits
> std::basic_ostream<_CharT, _Traits>& std::operator<<
(std::basic_ostream<_CharT, _Traits> & __os, const
std::independent_bits_engine<_RandomNumberEngine, __w,
_UIntType> & __x)

```

Inserts the current state of a `independent_bits_engine` random number generator engine `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `independent_bits_engine` random number generator engine.

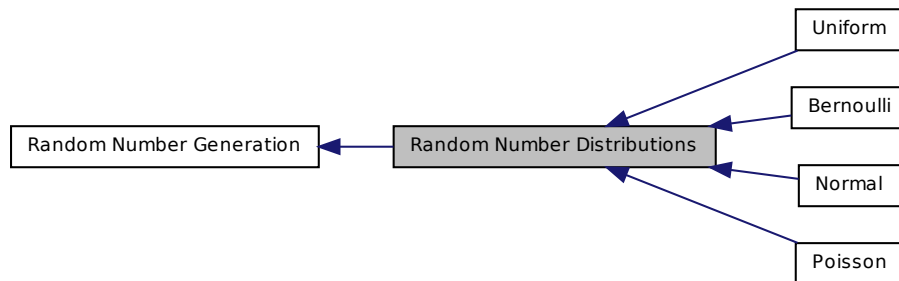
#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1219 of file `random.h`.

## 2.51 Random Number Distributions

Collaboration diagram for Random Number Distributions:



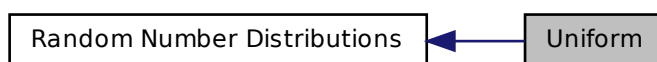
#### Modules

- [Uniform](#)

- [Normal](#)
- [Bernoulli](#)
- [Poisson](#)

## 2.52 Uniform

Collaboration diagram for Uniform:



### Classes

- class [std::uniform\\_int\\_distribution< \\_IntType >](#)  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- class [std::uniform\\_real\\_distribution< \\_RealType >](#)  
*Uniform continuous distribution for random numbers.*

### Functions

- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1,`  
`const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1,`  
`const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`  
`ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _`  
`RealType > &)`



- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType >  
bool std::operator==(const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator==(const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`

## 2.52.1 Function Documentation

**2.52.1.1** `template<typename _IntType > bool std::operator!=( const  
std::uniform_int_distribution< _IntType > & __d1, const  
std::uniform_int_distribution< _IntType > & __d2 ) [inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1758 of file random.h.

**2.52.1.2** `template<typename _IntType > bool std::operator!=( const  
std::uniform_real_distribution< _IntType > & __d1, const  
std::uniform_real_distribution< _IntType > & __d2 ) [inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 1939 of file random.h.

**2.52.1.3** `template<typename _RealType, typename _CharT, typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::uniform_real_distribution< _RealType > & __x )`

Inserts a uniform\_real\_distribution random number distribution \_\_x into the output stream \_\_os.

**Parameters**

`__os` An output stream.

`__x` A `uniform_real_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 934 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.52.1.4** `template<typename _IntType , typename _CharT , typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::uniform_int_distribution< _IntType > & __x )`

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

**Parameters**

`__os` An output stream.

`__x` A `uniform_int_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 892 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

**2.52.1.5** `template<typename _IntType > bool std::operator==( const  
std::uniform_real_distribution< _IntType > & __d1, const  
std::uniform_real_distribution< _IntType > & __d2 ) [inline]`

Return true if two uniform real distributions have the same parameters.

Definition at line 1929 of file `random.h`.

References `std::uniform_real_distribution< _RealType >::param()`.

**2.52.1.6** `template<typename _IntType > bool std::operator==( const  
std::uniform_int_distribution< _IntType > & __d1, const  
std::uniform_int_distribution< _IntType > & __d2 ) [inline]`

Return true if two uniform integer distributions have the same parameters.

Definition at line 1748 of file random.h.

References `std::uniform_int_distribution<_IntType>::param()`.

**2.52.1.7** `template<typename _RealType , typename _CharT ,  
typename _Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::uniform_real_distribution< _RealType > & __x )`

Extracts a `uniform_real_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `uniform_real_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 958 of file random.tcc.

References `std::ios_base::flags()`, `std::uniform_real_distribution<_RealType>::param()`, and `std::skipws()`.

**2.52.1.8** `template<typename _IntType , typename _CharT , typename  
_Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::uniform_int_distribution< _IntType > & __x )`

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `uniform_int_distribution` random number generator engine.

#### Returns

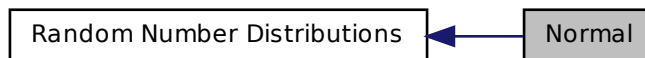
The input stream with `__x` extracted or in an error state.

Definition at line 913 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution<_IntType>::param()`, and `std::skipws()`.

## 2.53 Normal

Collaboration diagram for Normal:



### Classes

- class `std::cauchy_distribution<_RealType>`  
A *cauchy\_distribution* random number distribution.
- class `std::chi_squared_distribution<_RealType>`  
A *chi\_squared\_distribution* random number distribution.
- class `std::fisher_f_distribution<_RealType>`  
A *fisher\_f\_distribution* random number distribution.
- class `std::gamma_distribution<_RealType>`  
A *gamma* continuous distribution for random numbers.
- class `std::lognormal_distribution<_RealType>`  
A *lognormal\_distribution* random number distribution.
- class `std::normal_distribution<_RealType>`  
A *normal* continuous distribution for random numbers.
- class `std::student_t_distribution<_RealType>`  
A *student\_t\_distribution* random number distribution.

### Functions

- template<typename \_RealType>  
bool `std::operator!=` (const `std::normal_distribution<_RealType>` &\_\_d1,  
const `std::normal_distribution<_RealType>` &\_\_d2)

- `template<typename _RealType >`  
`bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1,`  
`const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`  
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`  
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`  
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1,`  
`const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType >`  
`&)`
- `template<typename _RealType >`  
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

### 2.53.1 Function Documentation

**2.53.1.1** `template<typename _RealType > bool std::operator!= ( const`  
`std::normal_distribution< _RealType > & __d1, const`  
`std::normal_distribution< _RealType > & __d2 ) [inline]`

Return true if two normal distributions are different.

Definition at line 2159 of file random.h.

**2.53.1.2** `template<typename _RealType > bool std::operator!= ( const`  
`std::lognormal_distribution< _RealType > & __d1, const`  
`std::lognormal_distribution< _RealType > & __d2 ) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2335 of file random.h.

**2.53.1.3** `template<typename _RealType > bool std::operator!= ( const  
std::chi_squared_distribution< _RealType > & __d1, const  
std::chi_squared_distribution< _RealType > & __d2 ) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2692 of file random.h.

**2.53.1.4** `template<typename _RealType > bool std::operator!= ( const  
std::fisher_f_distribution< _RealType > & __d1, const  
std::fisher_f_distribution< _RealType > & __d2 ) [inline]`

Return true if two Fisher f distributions are diferent.

Definition at line 3047 of file random.h.

**2.53.1.5** `template<typename _RealType > bool std::operator!= ( const  
std::student_t_distribution< _RealType > & __d1, const  
std::student_t_distribution< _RealType > & __d2 ) [inline]`

Return true if two Student t distributions are different.

Definition at line 3221 of file random.h.

**2.53.1.6** `template<typename _RealType > bool std::operator!= ( const  
std::cauchy_distribution< _RealType > & __d1, const  
std::cauchy_distribution< _RealType > & __d2 ) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 2831 of file random.h.

**2.53.1.7** `template<typename _RealType > bool std::operator!= ( const  
std::gamma_distribution< _RealType > & __d1, const  
std::gamma_distribution< _RealType > & __d2 ) [inline]`

Return true if two gamma distributions are different.

Definition at line 2527 of file random.h.

**2.53.1.8** `template<typename _RealType, typename _CharT, typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::cauchy_distribution< _RealType > & __x )`

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `cauchy_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1856 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

**2.53.1.9** `template<typename _RealType > bool std::operator==(  
const std::cauchy_distribution< _RealType > & __d1, const  
std::cauchy_distribution< _RealType > & __d2 ) [inline]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 2821 of file `random.h`.

References `std::cauchy_distribution< _RealType >::param()`.

**2.53.1.10** `template<typename _RealType, typename _CharT,  
typename _Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::cauchy_distribution< _RealType > & __x )`

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `cauchy_distribution` random number generator engine.

#### Returns

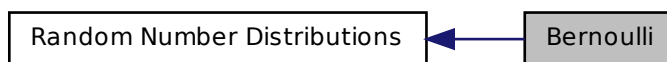
The input stream with `__x` extracted or in an error state.

Definition at line 1880 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType >::param()`, and `std::skipws()`.

## 2.54 Bernoulli

Collaboration diagram for Bernoulli:



### Classes

- class [std::bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- class [std::binomial\\_distribution< \\_IntType >](#)  
*A discrete binomial random number distribution.*
- class [std::geometric\\_distribution< \\_IntType >](#)  
*A discrete geometric random number distribution.*
- class [std::negative\\_binomial\\_distribution< \\_IntType >](#)  
*A [negative\\_binomial\\_distribution](#) random number distribution.*

### Functions

- `bool std::operator!= (const std::bernoulli\_distribution &__d1, const std::bernoulli\_distribution &__d2)`
- `template<typename _IntType >  
bool std::operator!= (const std::binomial\_distribution< _IntType > &__d1, const std::binomial\_distribution< _IntType > &__d2)`



- `template<typename _IntType >`  
`bool std::operator!= (const std::negative_binomial_distribution< _IntType >`  
`&__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1,`  
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`  
`ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType >`  
`&)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`  
`ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `bool std::operator== (const std::bernoulli_distribution &__d1, const`  
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1,`  
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &__is, std::bernoulli_distribution &__x)`

## 2.54.1 Function Documentation

### 2.54.1.1 `bool std::operator!= ( const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2 ) [inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3369 of file random.h.

### 2.54.1.2 `template<typename _IntType > bool std::operator!= ( const std::binomial_distribution< _IntType > & __d1, const std::binomial_distribution< _IntType > & __d2 ) [inline]`

Return true if two binomial distributions are different.

Definition at line 3606 of file random.h.

**2.54.1.3** `template<typename _IntType > bool std::operator!= ( const  
std::negative_binomial_distribution< _IntType > & __d1, const  
std::negative_binomial_distribution< _IntType > & __d2 )  
[inline]`

Return true if two negative binomial distributions are different.

Definition at line 3952 of file random.h.

**2.54.1.4** `template<typename _IntType > bool std::operator!= ( const  
std::geometric_distribution< _IntType > & __d1, const  
std::geometric_distribution< _IntType > & __d2 ) [inline]`

Return true if two geometric distributions have different parameters.

Definition at line 3748 of file random.h.

**2.54.1.5** `template<typename _IntType , typename _CharT , typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::geometric_distribution< _IntType > & __x )`

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `geometric_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1029 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.54.1.6** `template<typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::bernoulli_distribution & __x )`

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

`__os` An output stream.  
`__x` A `bernoulli_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 979 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.54.1.7** `bool std::operator==( const std::bernoulli_distribution & __d1,  
const std::bernoulli_distribution & __d2 ) [inline]`

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3360 of file `random.h`.

References `std::bernoulli_distribution::param()`.

**2.54.1.8** `template<typename _IntType > bool std::operator==( const  
std::geometric_distribution< _IntType > & __d1, const  
std::geometric_distribution< _IntType > & __d2 ) [inline]`

Return true if two geometric distributions have the same parameters.

Definition at line 3738 of file `random.h`.

References `std::geometric_distribution< _IntType >::param()`.

**2.54.1.9** `template<typename _IntType , typename _CharT , typename  
_Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>>( std::basic_istream< _CharT, _Traits > & __is,  
std::geometric_distribution< _IntType > & __x )`

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

`__is` An input stream.  
`__x` A `geometric_distribution` random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

Definition at line 1053 of file random.tcc.

References `std::ios_base::flags()`, `std::geometric_distribution< _IntType >::param()`, and `std::skipws()`.

```
2.54.1.10 template<typename _CharT , typename _Traits >
 std::basic_istream<_CharT, _Traits>& std::operator>>
 (std::basic_istream< _CharT, _Traits > & __is,
 std::bernoulli_distribution & __x)
```

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `bernoulli_distribution` random number generator engine.

#### Returns

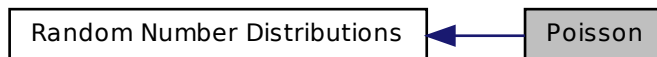
The input stream with `__x` extracted or in an error state.

Definition at line 3399 of file random.h.

References `std::bernoulli_distribution::param()`.

## 2.55 Poisson

Collaboration diagram for Poisson:



#### Classes

- class `std::discrete_distribution< _IntType >`

A *discrete\_distribution* random number distribution.

- class `std::exponential_distribution<_RealType>`  
A *exponential continuous distribution* for random numbers.
- class `std::extreme_value_distribution<_RealType>`  
A *extreme\_value\_distribution* random number distribution.
- class `std::piecewise_constant_distribution<_RealType>`  
A *piecewise\_constant\_distribution* random number distribution.
- class `std::piecewise_linear_distribution<_RealType>`  
A *piecewise\_linear\_distribution* random number distribution.
- class `std::poisson_distribution<_IntType>`  
A *discrete Poisson* random number distribution.
- class `std::weibull_distribution<_RealType>`  
A *weibull\_distribution* random number distribution.

## Functions

- template<typename \_IntType>  
bool `std::operator!=` (const `std::poisson_distribution<_IntType>` &\_\_d1, const `std::poisson_distribution<_IntType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::extreme_value_distribution<_RealType>` &\_\_d1, const `std::extreme_value_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::piecewise_constant_distribution<_RealType>` &\_\_d1, const `std::piecewise_constant_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::piecewise_linear_distribution<_RealType>` &\_\_d1, const `std::piecewise_linear_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::exponential_distribution<_RealType>` &\_\_d1, const `std::exponential_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::weibull_distribution<_RealType>` &\_\_d1, const `std::weibull_distribution<_RealType>` &\_\_d2)
- template<typename \_IntType>  
bool `std::operator!=` (const `std::discrete_distribution<_IntType>` &\_\_d1, const `std::discrete_distribution<_IntType>` &\_\_d2)

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _RealType >`  
`bool std::operator== (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`

### 2.55.1 Function Documentation

**2.55.1.1** `template<typename _IntType > bool std::operator!= ( const  
std::poisson_distribution< _IntType > & __d1, const  
std::poisson_distribution< _IntType > & __d2 ) [inline]`

Return true if two Poisson distributions are different.

Definition at line 4140 of file random.h.

**2.55.1.2** `template<typename _RealType > bool std::operator!= ( const  
std::extreme_value_distribution< _RealType > & __d1, const  
std::extreme_value_distribution< _RealType > & __d2 ) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 4640 of file random.h.

**2.55.1.3** `template<typename _RealType > bool std::operator!= ( const  
std::piecewise_constant_distribution< _RealType > & __d1, const  
std::piecewise_constant_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5070 of file random.h.

**2.55.1.4** `template<typename _RealType > bool std::operator!= ( const  
std::piecewise_linear_distribution< _RealType > & __d1, const  
std::piecewise_linear_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 5281 of file random.h.

**2.55.1.5** `template<typename _RealType > bool std::operator!= ( const  
std::exponential_distribution< _RealType > & __d1, const  
std::exponential_distribution< _RealType > & __d2 ) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4290 of file random.h.

**2.55.1.6** `template<typename _RealType > bool std::operator!=(  
const std::weibull_distribution< _RealType > & __d1, const  
std::weibull_distribution< _RealType > & __d2 ) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 4465 of file random.h.

**2.55.1.7** `template<typename _IntType > bool std::operator!=( const  
std::discrete_distribution< _IntType > & __d1, const  
std::discrete_distribution< _IntType > & __d2 ) [inline]`

Return true if two discrete distributions have different parameters.

Definition at line 4862 of file random.h.

**2.55.1.8** `template<typename _RealType , typename _CharT , typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::weibull_distribution< _RealType > & __x )`

Inserts a weibull\_distribution random number distribution \_\_x into the output stream \_\_os.

#### Parameters

`__os` An output stream.

`__x` A weibull\_distribution random number distribution.

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

Definition at line 2109 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.55.1.9** `template<typename _RealType , typename _CharT , typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::exponential_distribution< _RealType > & __x )`

Inserts a exponential\_distribution random number distribution \_\_x into the output stream \_\_os.



**Parameters**

`__os` An output stream.

`__x` A exponential\_distribution random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1589 of file random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.55.1.10** `template<typename _RealType , typename _CharT , typename  
_Traits > std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::extreme_value_distribution< _RealType > & __x )`

Inserts a extreme\_value\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

`__os` An output stream.

`__x` A extreme\_value\_distribution random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2166 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.55.1.11** `template<typename _RealType > bool std::operator==( const  
std::exponential_distribution< _RealType > & __d1, const  
std::exponential_distribution< _RealType > & __d2 ) [inline]`

Return true if two exponential distributions have the same parameters.

Definition at line 4280 of file random.h.

References `std::exponential_distribution< _RealType >::param()`.

**2.55.1.12** `template<typename _RealType > bool std::operator==( const  
std::piecewise_constant_distribution< _RealType > & __d1, const  
std::piecewise_constant_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5060 of file random.h.

References `std::piecewise_constant_distribution< _RealType >::param()`.

**2.55.1.13** `template<typename _RealType > bool std::operator==( const  
std::piecewise_linear_distribution< _RealType > & __d1, const  
std::piecewise_linear_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5271 of file random.h.

References `std::piecewise_linear_distribution< _RealType >::param()`.

**2.55.1.14** `template<typename _IntType > bool std::operator==( const  
std::discrete_distribution< _IntType > & __d1, const  
std::discrete_distribution< _IntType > & __d2 ) [inline]`

Return true if two discrete distributions have the same parameters.

Definition at line 4852 of file random.h.

References `std::discrete_distribution< _IntType >::param()`.

**2.55.1.15** `template<typename _RealType > bool std::operator==( const  
std::extreme_value_distribution< _RealType > & __d1, const  
std::extreme_value_distribution< _RealType > & __d2 )  
[inline]`

Return true if two extreme value distributions have the same parameters.

Definition at line 4630 of file random.h.

References `std::extreme_value_distribution< _RealType >::param()`.

**2.55.1.16** `template<typename _RealType > bool std::operator==( const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2 ) [inline]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4455 of file random.h.

References `std::weibull_distribution< _RealType >::param()`.

**2.55.1.17** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::exponential_distribution< _RealType > & __x )`

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `exponential_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1612 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution< _RealType >::param()`, and `std::skipws()`.

**2.55.1.18** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::weibull_distribution< _RealType > & __x )`

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `weibull_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2133 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution< _RealType >::param()`, and `std::skipws()`.

**2.55.1.19** `template<typename _RealType , typename _CharT ,  
typename _Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::extreme_value_distribution< _RealType > & __x )`

Extracts a `extreme_value_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `extreme_value_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2190 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution< _RealType >::param()`, and `std::skipws()`.

## 2.56 Random Number Utilities

Collaboration diagram for Random Number Utilities:



#### Classes

- class [std::seed\\_seq](#)

---

The [\*seed\\_seq\*](#) class generates sequences of seeds for random number generators.



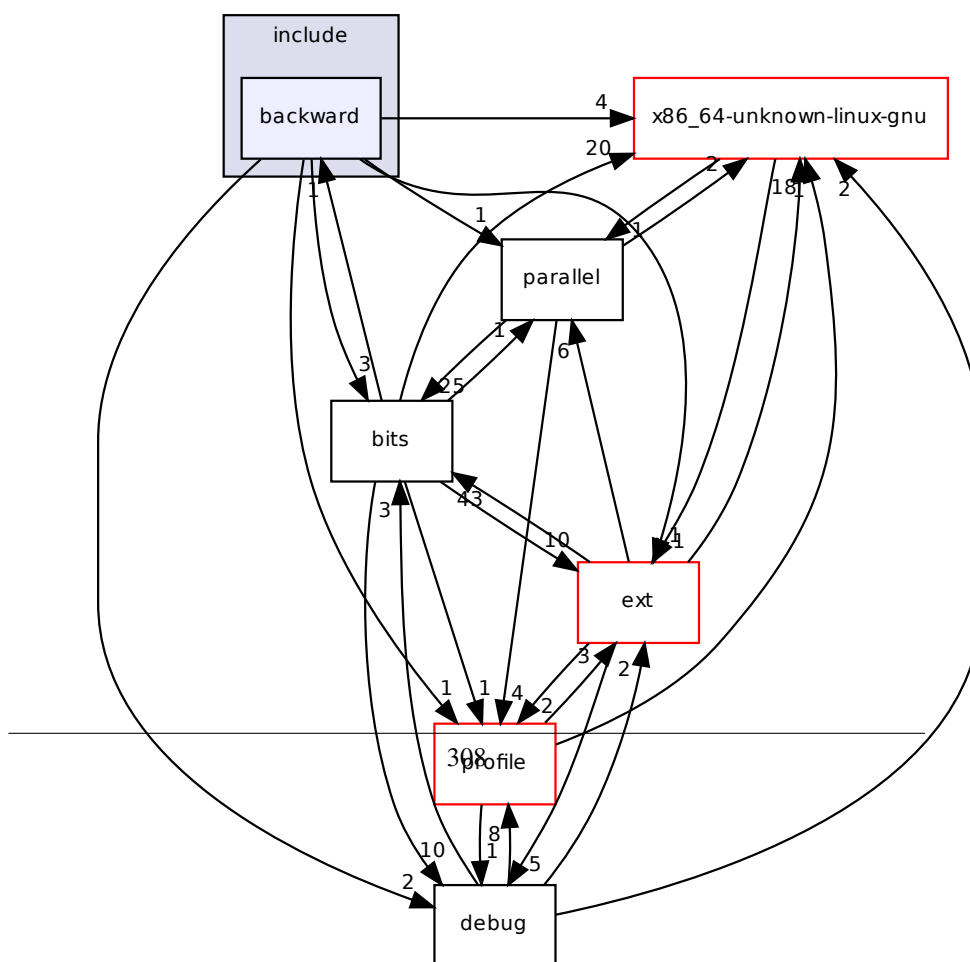


## Chapter 3

# Directory Documentation

### 3.1 include/backward/ Directory Reference

Directory dependency graph for include/backward/:



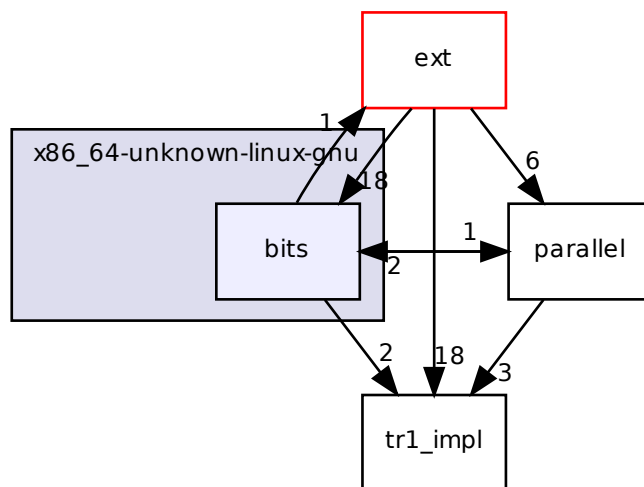


## Files

- file [auto\\_ptr.h](#)
- file **backward\_warning.h**
- file [binders.h](#)
- file [hash\\_fun.h](#)
- file [hash\\_map](#)
- file [hash\\_set](#)
- file [backward/hashtable.h](#)
- file **strstream**

## 3.2 include/x86\_64-unknown-linux-gnu/bits/ Directory Reference

Directory dependency graph for include/x86\_64-unknown-linux-gnu/bits/:



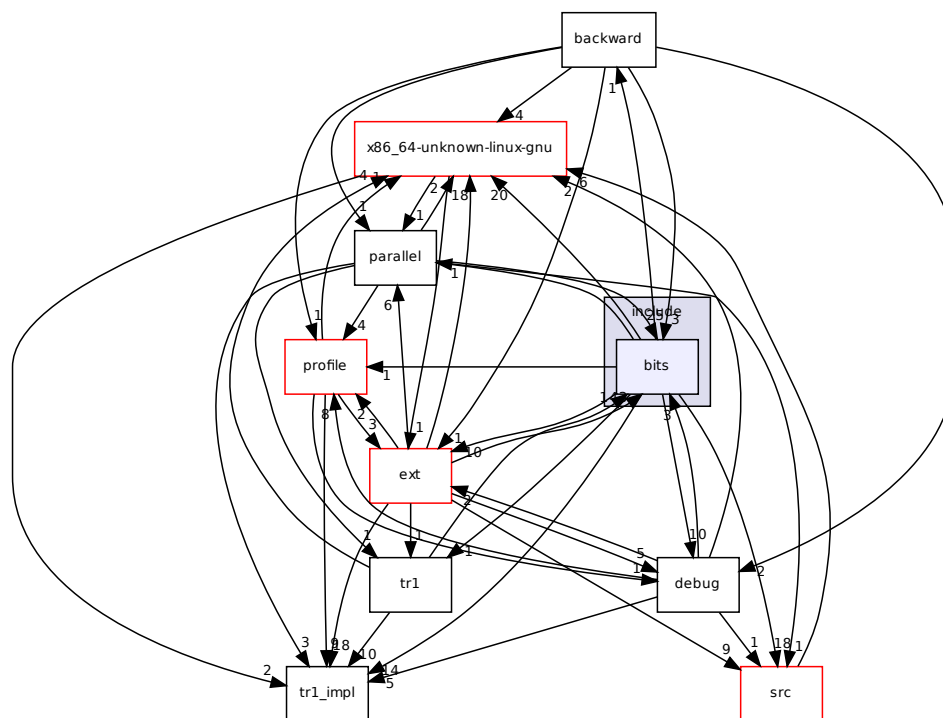
## Files

- file [atomic\\_word.h](#)

- file [basic\\_file.h](#)
- file [c++allocator.h](#)
- file [c++config.h](#)
- file [c++io.h](#)
- file [c++locale.h](#)
- file [c++locale\\_internal.h](#)
- file [x86\\_64-unknown-linux-gnu/bits/compatibility.h](#)
- file [cpu\\_defines.h](#)
- file [ctype\\_base.h](#)
- file [ctype\\_inline.h](#)
- file [ctype\\_noninline.h](#)
- file [cxxabi\\_tweaks.h](#)
- file [error\\_constants.h](#)
- file **gthr-default.h**
- file **gthr-posix.h**
- file **gthr-single.h**
- file **gthr-tpf.h**
- file **gthr.h**
- file [messages\\_members.h](#)
- file [os\\_defines.h](#)
- file [time\\_members.h](#)

### 3.3 include/bits/ Directory Reference

Directory dependency graph for include/bits/:



#### Files

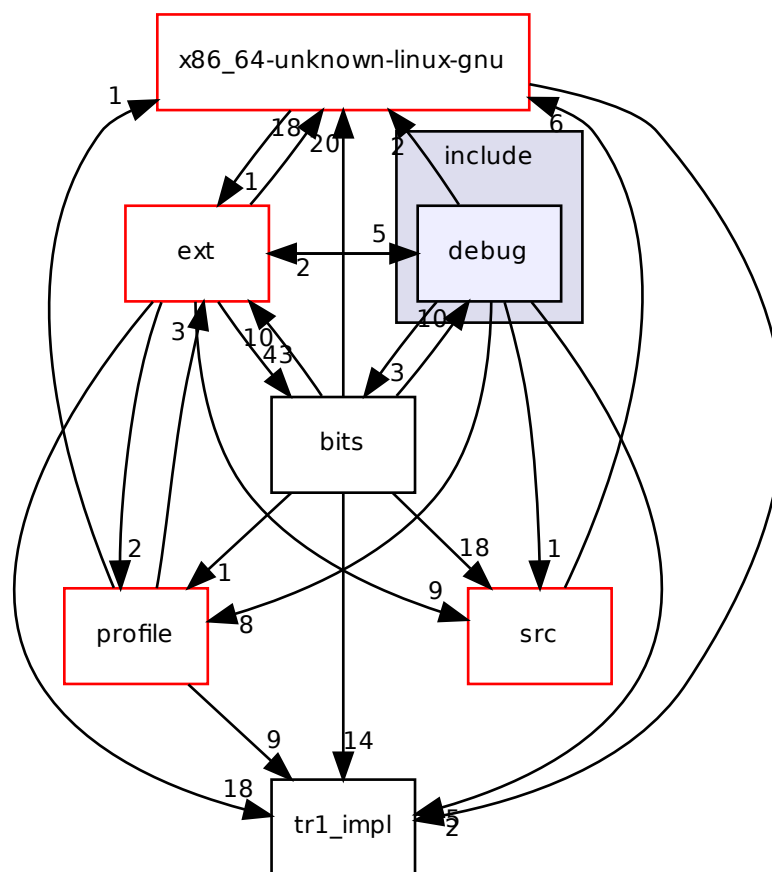
- file [bits/algorithmfwd.h](#)
- file [allocator.h](#)
- file [atomic\\_0.h](#)
- file [atomic\\_2.h](#)
- file [atomic\\_base.h](#)
- file [atomicfwd\\_c.h](#)
- file [atomicfwd\\_cxx.h](#)
- file [basic\\_ios.h](#)
- file [basic\\_ios.tcc](#)
- file [basic\\_string.h](#)

- file [basic\\_string.tcc](#)
- file [boost\\_concept\\_check.h](#)
- file [c++0x\\_warning.h](#)
- file [char\\_traits.h](#)
- file [cmath.tcc](#)
- file [codecvt.h](#)
- file [concept\\_check.h](#)
- file [cpp\\_type\\_traits.h](#)
- file [deque.tcc](#)
- file [forward\\_list.h](#)
- file [forward\\_list.tcc](#)
- file [fstream.tcc](#)
- file [functexcept.h](#)
- file [functional\\_hash.h](#)
- file [gslice.h](#)
- file [gslice\\_array.h](#)
- file [bits/hashtable.h](#)
- file [hashtable\\_policy.h](#)
- file [indirect\\_array.h](#)
- file [ios\\_base.h](#)
- file [istream.tcc](#)
- file [list.tcc](#)
- file [locale\\_classes.h](#)
- file [locale\\_classes.tcc](#)
- file [locale\\_facets.h](#)
- file [locale\\_facets.tcc](#)
- file [locale\\_facets\\_nonio.h](#)
- file [locale\\_facets\\_nonio.tcc](#)
- file [localefwd.h](#)
- file [mask\\_array.h](#)
- file [move.h](#)
- file [ostream.tcc](#)
- file [ostream\\_insert.h](#)
- file [postypes.h](#)
- file [random.h](#)
- file [random.tcc](#)
- file [regex.h](#)
- file [regex\\_compiler.h](#)
- file [regex\\_constants.h](#)
- file [regex\\_cursor.h](#)
- file [regex\\_error.h](#)
- file [regex\\_grep\\_matcher.h](#)
- file [regex\\_grep\\_matcher.tcc](#)

- file [regex\\_nfa.h](#)
- file [regex\\_nfa.tcc](#)
- file [shared\\_ptr.h](#)
- file [shared\\_ptr\\_base.h](#)
- file [slice\\_array.h](#)
- file [sstream.tcc](#)
- file [stl\\_algo.h](#)
- file [stl\\_algobase.h](#)
- file [stl\\_bvector.h](#)
- file [stl\\_construct.h](#)
- file [stl\\_deque.h](#)
- file [stl\\_function.h](#)
- file [stl\\_heap.h](#)
- file [stl\\_iterator.h](#)
- file [stl\\_iterator\\_base\\_funcs.h](#)
- file [stl\\_iterator\\_base\\_types.h](#)
- file [stl\\_list.h](#)
- file [stl\\_map.h](#)
- file [stl\\_multimap.h](#)
- file [stl\\_multiset.h](#)
- file [stl\\_numeric.h](#)
- file [stl\\_pair.h](#)
- file [stl\\_queue.h](#)
- file [stl\\_raw\\_storage\\_iter.h](#)
- file [stl\\_relops.h](#)
- file [stl\\_set.h](#)
- file [stl\\_stack.h](#)
- file [stl\\_tempbuf.h](#)
- file [stl\\_tree.h](#)
- file [stl\\_uninitialized.h](#)
- file [stl\\_vector.h](#)
- file [stream\\_iterator.h](#)
- file [streambuf.tcc](#)
- file [streambuf\\_iterator.h](#)
- file [stringfwd.h](#)
- file [unique\\_ptr.h](#)
- file [unordered\\_map.h](#)
- file [unordered\\_set.h](#)
- file [valarray\\_after.h](#)
- file [valarray\\_array.h](#)
- file [valarray\\_array.tcc](#)
- file [valarray\\_before.h](#)
- file [vector.tcc](#)

### 3.4 include/debug/ Directory Reference

Directory dependency graph for include/debug/:



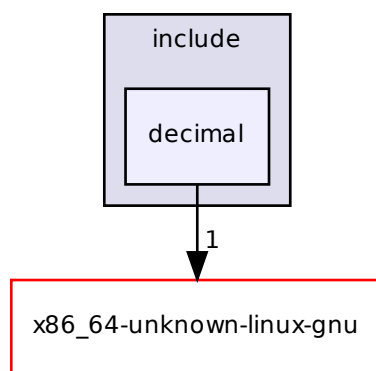
#### Files

- file [debug/bitset](#)
- file [debug.h](#)
- file [debug/deque](#)

- file [formatter.h](#)
- file [functions.h](#)
- file [debug/list](#)
- file [macros.h](#)
- file [debug/map](#)
- file [debug/map.h](#)
- file [debug/multimap.h](#)
- file [debug/multiset.h](#)
- file [safe\\_base.h](#)
- file [safe\\_iterator.h](#)
- file [safe\\_iterator.tcc](#)
- file [safe\\_sequence.h](#)
- file [debug/set](#)
- file [debug/set.h](#)
- file [debug/string](#)
- file [debug/unordered\\_map](#)
- file [debug/unordered\\_set](#)
- file [debug/vector](#)

### 3.5 include/decimal/ Directory Reference

Directory dependency graph for include/decimal/:

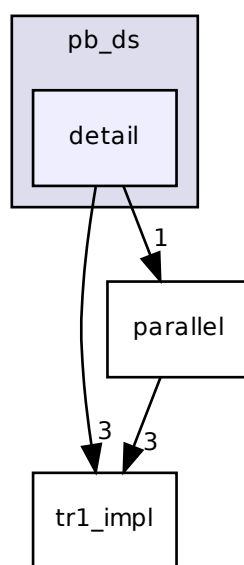


## Files

- file [decimal](#)

## 3.6 include/ext/pb\_ds/detail/ Directory Reference

Directory dependency graph for include/ext/pb\_ds/detail/:



## Files

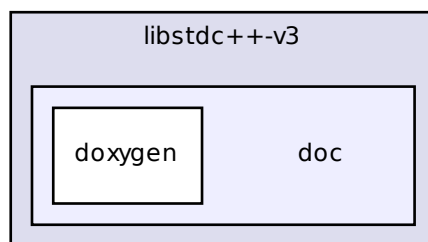
- file [basic\\_types.hpp](#)
- file [cond\\_dealtor.hpp](#)
- file [constructors\\_destructor\\_fn\\_imps.hpp](#)
- file [container\\_base\\_dispatch.hpp](#)
- file [debug\\_map\\_base.hpp](#)
- file [priority\\_queue\\_base\\_dispatch.hpp](#)



- file [standard\\_policies.hpp](#)
- file [tree\\_trace\\_base.hpp](#)
- file [type\\_utils.hpp](#)
- file [types\\_traits.hpp](#)

### 3.7 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/:

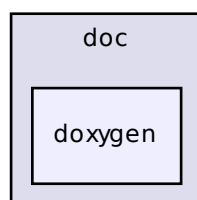


#### Directories

- directory [doxygen](#)

### 3.8 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/:

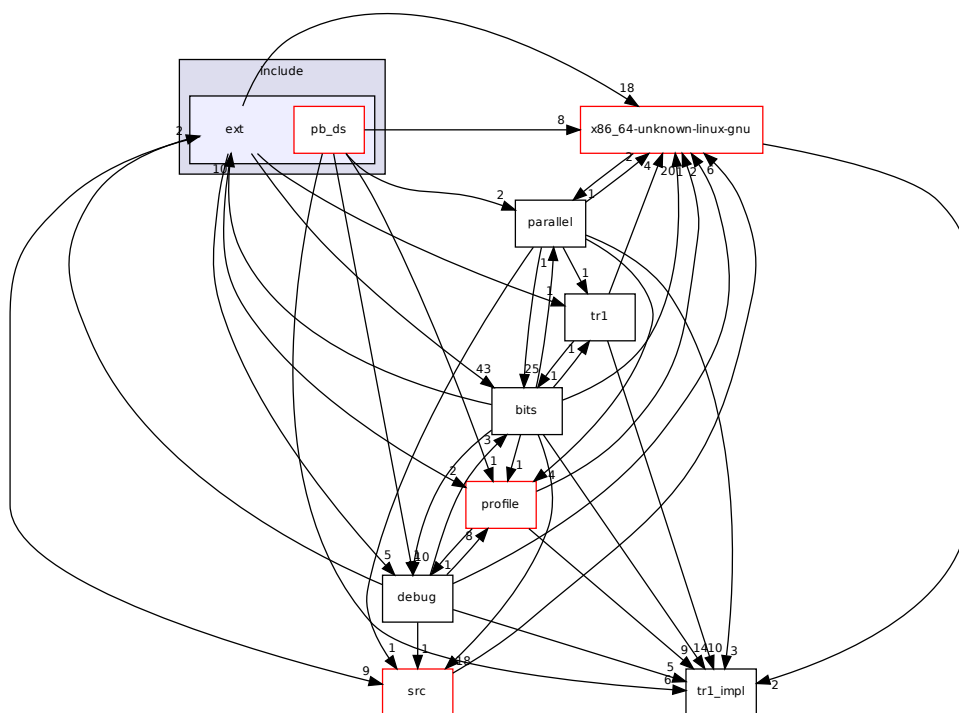


#### Files

- file **doxygroups.cc**

### 3.9 include/ext/ Directory Reference

Directory dependency graph for include/ext/:



#### Directories

- directory [pb\\_ds](#)

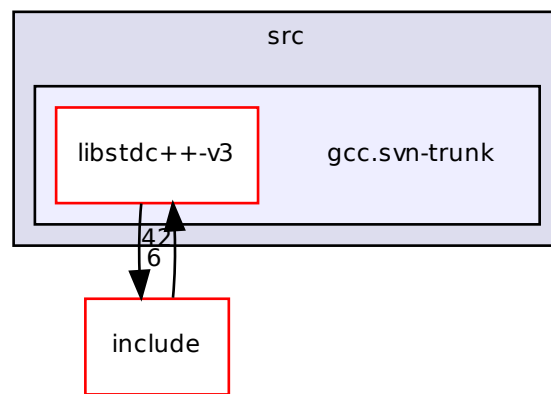
#### Files

- file [ext/algorithm](#)
- file [array\\_allocator.h](#)
- file [atomicity.h](#)
- file [bitmap\\_allocator.h](#)
- file [cast.h](#)
- file [codecvt\\_specializations.h](#)

- file [concurrency.h](#)
- file [debug\\_allocator.h](#)
- file [enc\\_filebuf.h](#)
- file [extptr\\_allocator.h](#)
- file [ext/functional](#)
- file [ext/iterator](#)
- file [malloc\\_allocator.h](#)
- file [ext/memory](#)
- file [mt\\_allocator.h](#)
- file [new\\_allocator.h](#)
- file [ext/numeric](#)
- file [numeric\\_traits.h](#)
- file [pod\\_char\\_traits.h](#)
- file [pointer.h](#)
- file [pool\\_allocator.h](#)
- file [rb\\_tree](#)
- file [rc\\_string\\_base.h](#)
- file [rope](#)
- file [ropeimpl.h](#)
- file [slist](#)
- file [sso\\_string\\_base.h](#)
- file [stdio\\_filebuf.h](#)
- file [stdio\\_sync\\_filebuf.h](#)
- file **[string\\_conversions.h](#)**
- file [throw\\_allocator.h](#)
- file [type\\_traits.h](#)
- file [typelist.h](#)
- file [vstring.h](#)
- file [vstring.tcc](#)
- file [vstring\\_fwd.h](#)
- file [vstring\\_util.h](#)

### 3.10 /mnt/share/src/gcc.svn-trunk/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/:

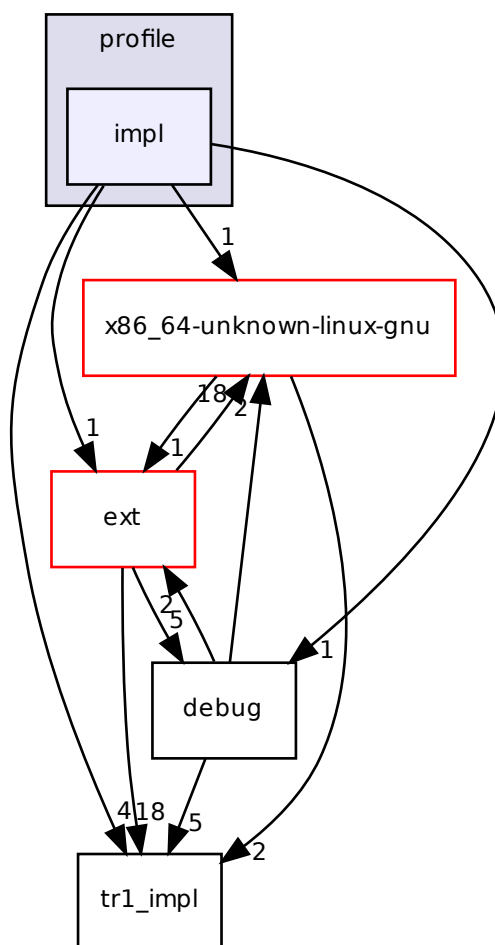


#### Directories

- directory [libstdc++-v3](#)

### 3.11 include/profile/impl/ Directory Reference

Directory dependency graph for include/profile/impl/:

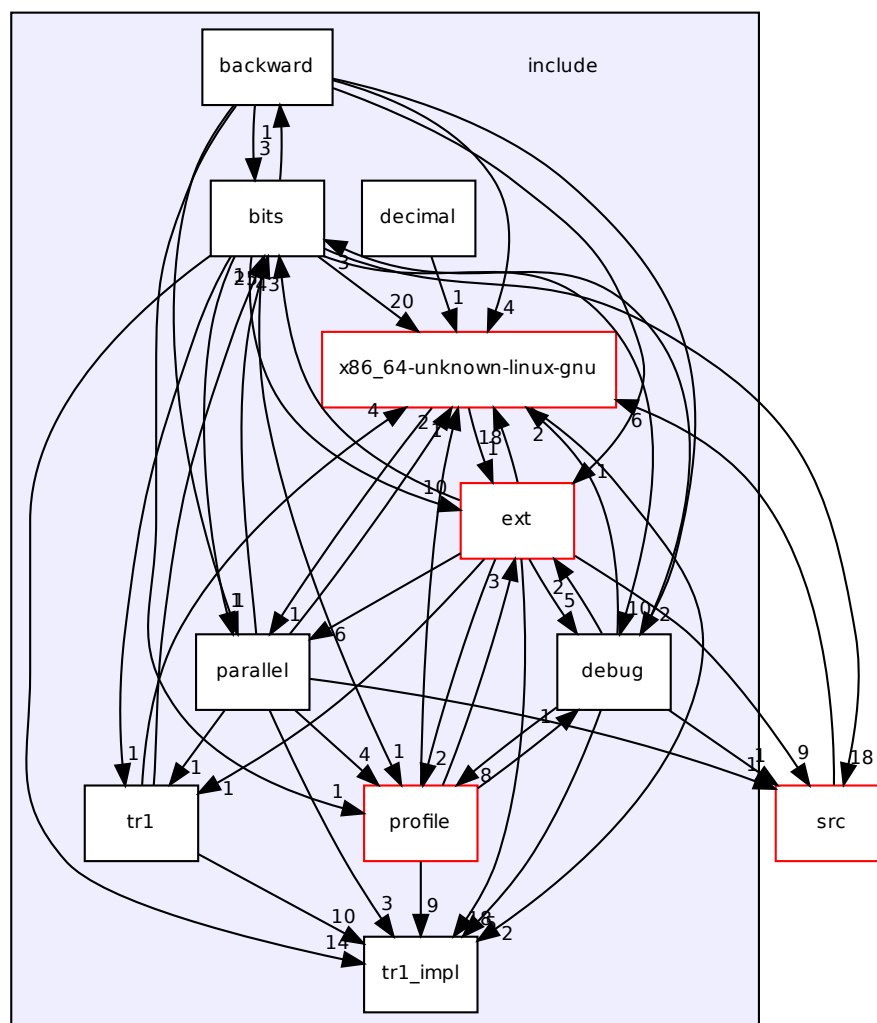


## Files

- file [profiler.h](#)
- file [profiler\\_algos.h](#)
- file **profiler\_container\_size.h**
- file **profiler\_hash\_func.h**
- file [profiler\\_hashtable\\_size.h](#)
- file [profiler\\_list\\_to\\_slist.h](#)
- file [profiler\\_list\\_to\\_vector.h](#)
- file [profiler\\_map\\_to\\_unordered\\_map.h](#)
- file [profiler\\_node.h](#)
- file [profiler\\_state.h](#)
- file [profiler\\_trace.h](#)
- file [profiler\\_vector\\_size.h](#)
- file [profiler\\_vector\\_to\\_list.h](#)

### 3.12 include/ Directory Reference

Directory dependency graph for include/:





## Directories

- directory [backward](#)
- directory [bits](#)
- directory [debug](#)
- directory [decimal](#)
- directory [ext](#)
- directory [parallel](#)
- directory [profile](#)
- directory [tr1](#)
- directory [tr1\\_impl](#)
- directory [x86\\_64-unknown-linux-gnu](#)

## Files

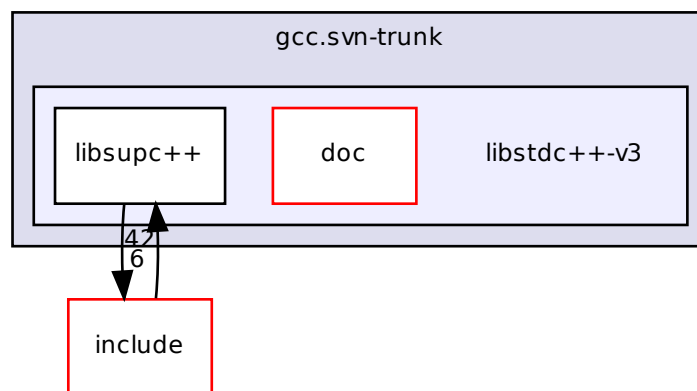
- file [algorithm](#)
- file [array](#)
- file [atomic](#)
- file [bitset](#)
- file [cassert](#)
- file [ccomplex](#)
- file [cctype](#)
- file [cerrno](#)
- file [cfenv](#)
- file [cfloat](#)
- file [chrono](#)
- file [cinttypes](#)
- file [ciso646](#)
- file [climits](#)
- file [clocale](#)
- file [cmath](#)
- file [complex](#)
- file [complex.h](#)
- file [condition\\_variable](#)
- file [csetjmp](#)
- file [csignal](#)
- file [cstdarg](#)
- file [cstdbool](#)
- file [cstddef](#)
- file [cstdint](#)
- file [cstdio](#)
- file [cstdlib](#)

- file [cstring](#)
- file [ctgmath](#)
- file [ctime](#)
- file [cwchar](#)
- file [cwctype](#)
- file [deque](#)
- file [fenv.h](#)
- file [fstream](#)
- file [functional](#)
- file [future](#)
- file **gstdint.h**
- file [iomanip](#)
- file [ios](#)
- file [iosfwd](#)
- file [iostream](#)
- file [istream](#)
- file [iterator](#)
- file [limits](#)
- file [list](#)
- file [locale](#)
- file [map](#)
- file [memory](#)
- file [mutex](#)
- file [numeric](#)
- file [ostream](#)
- file [queue](#)
- file [random](#)
- file [ratio](#)
- file [regex](#)
- file [set](#)
- file [sstream](#)
- file [stack](#)
- file [stdatomic.h](#)
- file [stdexcept](#)
- file [streambuf](#)
- file [string](#)
- file [system\\_error](#)
- file [tgmath.h](#)
- file [thread](#)
- file [tuple](#)
- file [type\\_traits](#)
- file [unordered\\_map](#)
- file [unordered\\_set](#)

- file [utility](#)
- file [valarray](#)
- file [vector](#)

### 3.13 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/:

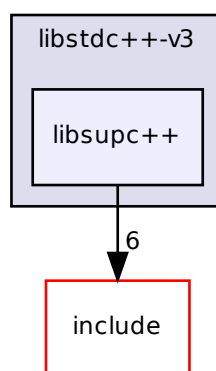


#### Directories

- directory [doc](#)
- directory [libsupc++](#)

### 3.14 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/:

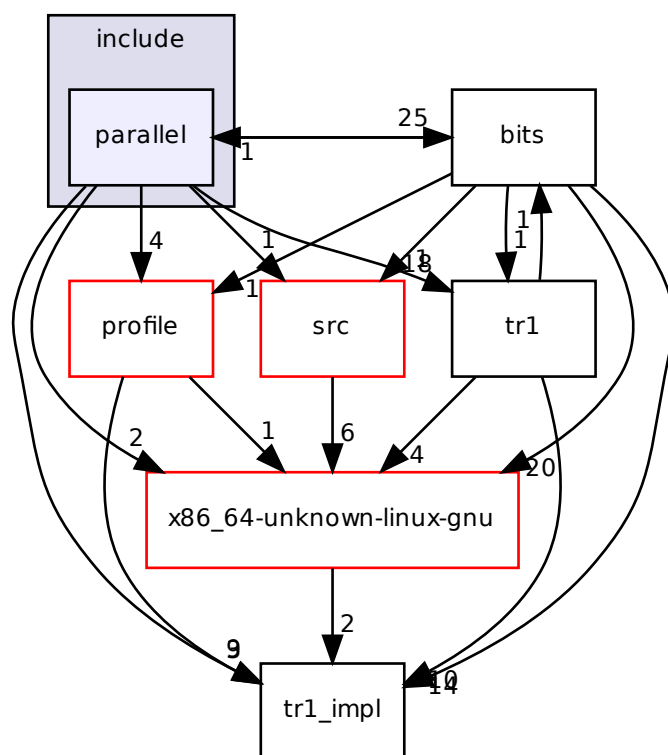


#### Files

- file [cxxabi-forced.h](#)
- file [cxxabi.h](#)
- file [exception](#)
- file [exception\\_ptr.h](#)
- file [initializer\\_list](#)
- file [nested\\_exception.h](#)
- file [new](#)
- file [typeinfo](#)

### 3.15 include/parallel/ Directory Reference

Directory dependency graph for include/parallel/:



## Files

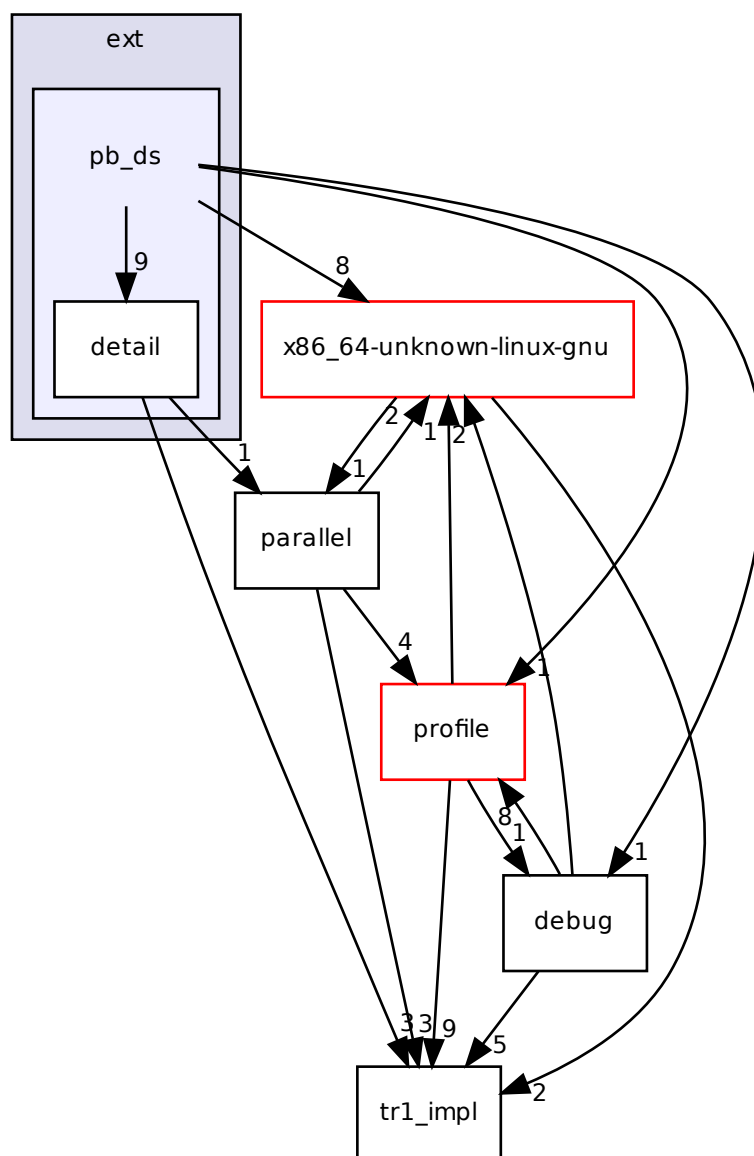
- file `algo.h`
- file `algotbase.h`
- file `parallel/algorithm`
- file `parallel/algorithmfwd.h`
- file `balanced_quicksort.h`
- file `parallel/base.h`

- file [basic\\_iterator.h](#)
- file [checkers.h](#)
- file [parallel/compatibility.h](#)
- file [compiletime\\_settings.h](#)
- file [equally\\_split.h](#)
- file [features.h](#)
- file [find.h](#)
- file [find\\_selectors.h](#)
- file [for\\_each.h](#)
- file [for\\_each\\_selectors.h](#)
- file [iterator.h](#)
- file [list\\_partition.h](#)
- file [losertree.h](#)
- file [merge.h](#)
- file [multiseq\\_selection.h](#)
- file [multiway\\_merge.h](#)
- file [multiway\\_mergesort.h](#)
- file [parallel/numeric](#)
- file [numericfwd.h](#)
- file [omp\\_loop.h](#)
- file [omp\\_loop\\_static.h](#)
- file [par\\_loop.h](#)
- file [parallel.h](#)
- file [partial\\_sum.h](#)
- file [partition.h](#)
- file [queue.h](#)
- file [quicksort.h](#)
- file [random\\_number.h](#)
- file [random\\_shuffle.h](#)
- file [search.h](#)
- file [set\\_operations.h](#)
- file [settings.h](#)
- file [sort.h](#)
- file [tags.h](#)
- file [types.h](#)
- file [unique\\_copy.h](#)
- file [workstealing.h](#)



### 3.16 include/ext/pb\_ds/ Directory Reference

Directory dependency graph for include/ext/pb\_ds/:





## Directories

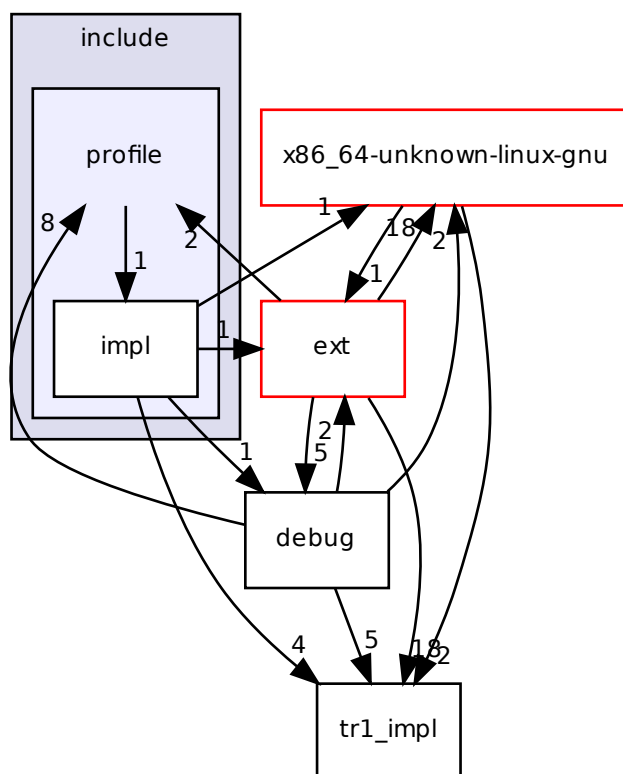
- directory [detail](#)

## Files

- file [assoc\\_container.hpp](#)
- file [exception.hpp](#)
- file [hash\\_policy.hpp](#)
- file [list\\_update\\_policy.hpp](#)
- file [priority\\_queue.hpp](#)
- file [tag\\_and\\_trait.hpp](#)
- file [tree\\_policy.hpp](#)
- file [trie\\_policy.hpp](#)

### 3.17 include/profile/ Directory Reference

Directory dependency graph for include/profile/:



#### Directories

- directory [impl](#)

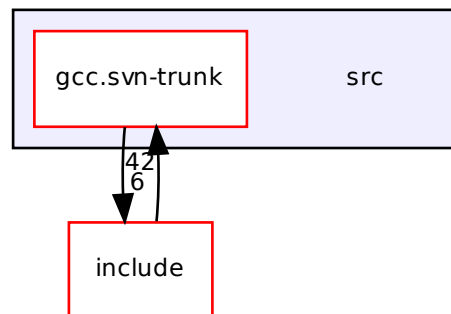
#### Files

- file [profile/base.h](#)

- file [profile/bitset](#)
- file [profile/deque](#)
- file [iterator\\_tracker.h](#)
- file [profile/list](#)
- file [profile/map](#)
- file [profile/map.h](#)
- file [profile/multimap.h](#)
- file [profile/multiset.h](#)
- file [profile/set](#)
- file [profile/set.h](#)
- file [profile/unordered\\_map](#)
- file [profile/unordered\\_set](#)
- file [profile/vector](#)

### 3.18 /mnt/share/src/ Directory Reference

Directory dependency graph for /mnt/share/src/:

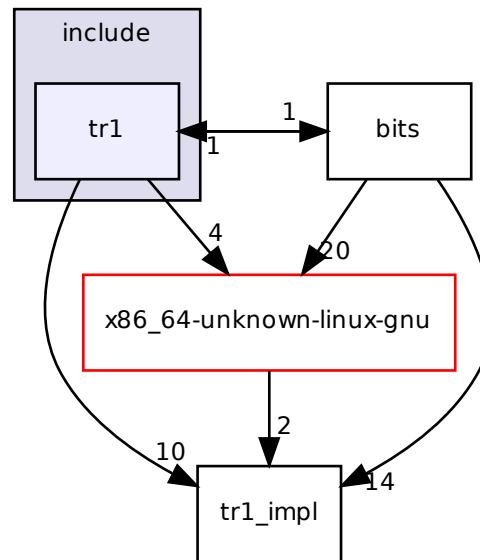


#### Directories

- directory [gcc.svn-trunk](#)

### 3.19 include/tr1/ Directory Reference

Directory dependency graph for include/tr1/:



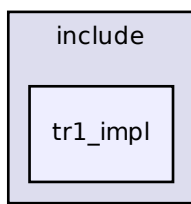
#### Files

- file [tr1/ccomplex](#)
- file [tr1/cctype](#)
- file [tr1/cfenv](#)
- file [tr1/cfloat](#)
- file [tr1/cinttypes](#)
- file [tr1/climits](#)
- file [tr1/cmath](#)
- file [tr1/complex](#)
- file [tr1/cstdarg](#)
- file [tr1/cstdbool](#)
- file [tr1/cstdint](#)

- file [tr1/cstdio](#)
- file [tr1/cstdlib](#)
- file [tr1/ctgmath](#)
- file [tr1/ctime](#)
- file [tr1/cwchar](#)
- file [tr1/cwctype](#)

## 3.20 include/tr1\_impl/ Directory Reference

Directory dependency graph for include/tr1\_impl/:

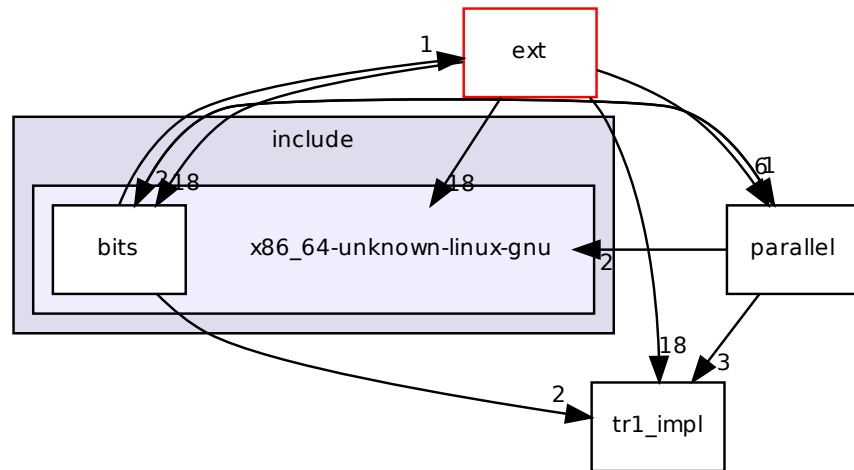


### Files

- file [tr1\\_impl/array](#)
- file [boost\\_sp\\_counted\\_base.h](#)
- file [tr1\\_impl/cctype](#)
- file [tr1\\_impl/cfenv](#)
- file [tr1\\_impl/cinttypes](#)
- file [tr1\\_impl/cmath](#)
- file [tr1\\_impl/complex](#)
- file [tr1\\_impl/cstdint](#)
- file [tr1\\_impl/cstdio](#)
- file [tr1\\_impl/cstdlib](#)
- file [tr1\\_impl/cwchar](#)
- file [tr1\\_impl/cwctype](#)
- file [tr1\\_impl/type\\_traits](#)
- file [tr1\\_impl/utility](#)

### 3.21 include/x86\_64-unknown-linux-gnu/ Directory Reference

Directory dependency graph for include/x86\_64-unknown-linux-gnu/:



#### Directories

- directory [bits](#)

## Chapter 4

# Namespace Documentation

### 4.1 `__gnu_cxx` Namespace Reference

GNU extensions for public use.

#### Namespaces

- namespace [\\_\\_detail](#)
- namespace [typelist](#)

#### Classes

- struct [\\_\\_common\\_pool\\_policy](#)  
*Policy for shared `__pool` objects.*
- class [\\_\\_mt\\_alloc](#)  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- class [\\_\\_mt\\_alloc\\_base](#)  
*Base class for `_Tp` dependent member functions.*
- struct [\\_\\_per\\_type\\_pool\\_policy](#)  
*Policy for individual `__pool` objects.*

- class [\\_\\_pool< false >](#)  
*Specialization for single thread.*
- class [\\_\\_pool< true >](#)  
*Specialization for thread enabled, via `gthreads.h`.*
- class [\\_\\_pool\\_alloc](#)  
*Allocator using a memory pool with a single lock.*
- class [\\_\\_pool\\_alloc\\_base](#)  
*Base class for [\\_\\_pool\\_alloc](#).*
- struct [\\_\\_pool\\_base](#)  
*Base class for pool object.*
- class [\\_\\_rc\\_string\\_base](#)
- class [\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*
- class [\\_\\_versa\\_string](#)  
*Template class [\\_\\_versa\\_string](#).  
Data structure managing sequences of characters and character-like objects.*
- struct [\\_Caster](#)
- struct [\\_Char\\_types](#)  
*Mapping from character type to associated types.*
- class [\\_ExtPtr\\_allocator](#)  
*An example allocator which uses a non-standard pointer type.  
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*
- struct [\\_Invalid\\_type](#)
- class [\\_Pointer\\_adapter](#)
- class [\\_Relative\\_pointer\\_impl](#)  
*A storage policy for use with [\\_Pointer\\_adapter<>](#) which stores the pointer's address as an offset value which is relative to its own address.*
- class [\\_Relative\\_pointer\\_impl< const \\_Tp >](#)
- class [\\_Std\\_pointer\\_impl](#)  
*A storage policy for use with [\\_Pointer\\_adapter<>](#) which yields a standard pointer.*



- struct `_Unqualified_type`
- struct `annotate_base`

*Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*
- class `array_allocator`

*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class `array_allocator_base`

*Base class.*
- class `binary_compose`

*An SGI extension .*
- class `bitmap_allocator`

*Bitmap Allocator, primary template.*
- struct `char_traits`

*Base class used to implement `std::char_traits`.*
- struct `character`

*A POD class that serves as a character abstraction class.*
- struct `condition_base`

*Base struct for condition policy.*
- struct `constant_binary_fun`

*An SGI extension .*
- struct `constant_unary_fun`

*An SGI extension .*
- struct `constant_void_fun`

*An SGI extension .*
- class `debug_allocator`

*A meta-allocator with debugging bits, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*

  - all allocation calls operator `new`
  - all deallocation calls operator `delete`.

- class [enc\\_filebuf](#)  
*class [enc\\_filebuf](#).*
- struct [encoding\\_char\\_traits](#)  
*[encoding\\_char\\_traits](#)*
- class [encoding\\_state](#)  
*Extension to use [iconv](#) for dealing with character encodings.*
- struct [forced\\_error](#)  
*Thrown by exception safety machinery.*
- class [free\\_list](#)  
*The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).*
- class [hash\\_map](#)
- class [hash\\_multimap](#)
- class [hash\\_multiset](#)
- class [hash\\_set](#)
- struct [limit\\_condition](#)  
*Base class for incremental control and throw.*
- class [malloc\\_allocator](#)  
*An allocator that uses [malloc](#).  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls [malloc](#)
  - all deallocation calls [free](#).
- class [new\\_allocator](#)  
*An allocator that uses global [new](#), as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls [operator new](#)
  - all deallocation calls [operator delete](#).
- struct [project1st](#)  
*An [SGI extension](#) .*
- struct [project2nd](#)  
*An [SGI extension](#) .*
- struct [random\\_condition](#)

*Base class for random probability control and throw.*

- struct `rb_tree`
- class `recursive_init_error`

*Exception thrown by `__cxa_guard_acquire`.  
6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `rope`
- struct `select1st`

*An SGI extension .*
- struct `select2nd`

*An SGI extension .*
- class `slist`
- class `stdio_filebuf`

*Provides a layer of compatibility for C/POSIX.  
This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `stdio_sync_filebuf`

*Provides a layer of compatibility for C.  
This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `subtractive_rng`
- struct `temporary_buffer`
- class `throw_allocator_base`

*Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.  
Note: Deallocate not allowed to throw.*
- struct `throw_allocator_limit`

*Allocator throwing via limit condition.*
- struct `throw_allocator_random`

*Allocator throwing via random condition.*
- struct `throw_value_base`

*Class with exception generation control. Intended to be used as a `value_type` in templated code.*

- struct [throw\\_value\\_limit](#)  
*Type throwing via limit condition.*
- struct [throw\\_value\\_random](#)  
*Type throwing via random condition.*
- class [unary\\_compose](#)  
*An SGI extension .*

## Typedefs

- typedef void(\* [\\_\\_destroy\\_handler](#))(void \*)
- typedef [\\_\\_versa\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_rc\\_string](#)
- typedef [\\_\\_vstring](#) [\\_\\_sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char16\_t, [std::char\\_traits](#)< char16\_t >, [std::allocator](#)< char16\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_u16rc\\_string](#)
- typedef [\\_\\_u16vstring](#) [\\_\\_u16sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char16\_t > [\\_\\_u16vstring](#)
- typedef [\\_\\_versa\\_string](#)< char32\_t, [std::char\\_traits](#)< char32\_t >, [std::allocator](#)< char32\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_u32rc\\_string](#)
- typedef [\\_\\_u32vstring](#) [\\_\\_u32sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char32\_t > [\\_\\_u32vstring](#)
- typedef [\\_\\_versa\\_string](#)< char > [\\_\\_vstring](#)
- typedef [\\_\\_versa\\_string](#)< wchar\_t, [std::char\\_traits](#)< wchar\_t >, [std::allocator](#)< wchar\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_wrc\\_string](#)
- typedef [\\_\\_wvstring](#) [\\_\\_wsso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< wchar\_t > [\\_\\_wvstring](#)
- typedef [rope](#)< char > [crope](#)
- typedef [rope](#)< wchar\_t > [wrope](#)

## Enumerations

- enum { [\\_S\\_num\\_primes](#) }
- enum [\\_Lock\\_policy](#) { [\\_S\\_single](#), [\\_S\\_mutex](#), [\\_S\\_atomic](#) }

## Functions

- static void **\_\_atomic\_add** (volatile `_Atomic_word` \* \_\_mem, int \_\_val)
- static void **\_\_atomic\_add\_single** (`_Atomic_word` \* \_\_mem, int \_\_val)
- static `_Atomic_word` **\_\_attribute\_\_** ((\_\_unused\_\_)) **\_\_exchange\_and\_add\_dispatch**(`_Atomic_word` \* \_\_mem
- template<class `_Tp` >  
void **\_\_aux\_require\_boolean\_expr** (const `_Tp` & \_\_t)
- template<typename `_ToType` , typename `_FromType` >  
`_ToType` **\_\_const\_pointer\_cast** (const `_FromType` & \_\_arg)
- template<typename `_ToType` , typename `_FromType` >  
`_ToType` **\_\_const\_pointer\_cast** (`_FromType` \* \_\_arg)
- template<typename `_InputIterator` , typename `_Size` , typename `_OutputIterator` >  
[pair](#)< `_InputIterator` , `_OutputIterator` > **\_\_copy\_n** (`_InputIterator` \_\_first, `_Size` \_\_count, `_OutputIterator` \_\_result, [input\\_iterator\\_tag](#))
- template<typename `_RAIterator` , typename `_Size` , typename `_OutputIterator` >  
[pair](#)< `_RAIterator` , `_OutputIterator` > **\_\_copy\_n** (`_RAIterator` \_\_first, `_Size` \_\_count, `_OutputIterator` \_\_result, [random\\_access\\_iterator\\_tag](#))
- template<typename `_InputIterator` , typename `_Distance` >  
void **\_\_distance** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `_Distance` & \_\_n, [std::input\\_iterator\\_tag](#))
- template<typename `_RandomAccessIterator` , typename `_Distance` >  
void **\_\_distance** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Distance` & \_\_n, [std::random\\_access\\_iterator\\_tag](#))
- template<typename `_ToType` , typename `_FromType` >  
`_ToType` **\_\_dynamic\_pointer\_cast** (const `_FromType` & \_\_arg)
- template<typename `_ToType` , typename `_FromType` >  
`_ToType` **\_\_dynamic\_pointer\_cast** (`_FromType` \* \_\_arg)
- void **\_\_error\_type\_must\_be\_a\_signed\_integer\_type** ()
- void **\_\_error\_type\_must\_be\_an\_integer\_type** ()
- void **\_\_error\_type\_must\_be\_an\_unsigned\_integer\_type** ()
- static `_Atomic_word` **\_\_exchange\_and\_add** (volatile `_Atomic_word` \* \_\_mem, int \_\_val)
- static `_Atomic_word` **\_\_exchange\_and\_add\_single** (`_Atomic_word` \* \_\_mem, int \_\_val)
- else return **\_\_exchange\_and\_add\_single** (\_\_mem, \_\_val)
- template<class `_Concept` >  
void **\_\_function\_requires** ()
- template<typename `_Type` >  
bool **\_\_is\_null\_pointer** (`_Type` \* \_\_ptr)
- template<typename `_Type` >  
bool **\_\_is\_null\_pointer** (`_Type`)
- int **\_\_lexicographical\_compare\_3way** (const unsigned char \* \_\_first1, const unsigned char \* \_\_last1, const unsigned char \* \_\_first2, const unsigned char \* \_\_last2)

- `int \_\_lexicographical\_compare\_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `template<typename _InputIterator1, typename _InputIterator2 >  
int \_\_lexicographical\_compare\_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp >  
const _Tp & \_\_median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >  
const _Tp & \_\_median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference \_\_mutable\_reference\_at (crope &__c, size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >  
_Tp \_\_power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >  
_Tp \_\_power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >  
_RandomAccessIterator \_\_random\_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >  
_RandomAccessIterator \_\_random\_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_reinterpret\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_reinterpret\_pointer\_cast (_FromType *__arg)`
- `_Slist_node_base * \_\_slist\_make\_link (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)`
- `_Slist_node_base * \_\_slist\_previous (_Slist_node_base *__head, const _Slist_node_base *__node)`
- `const _Slist_node_base * \_\_slist\_previous (const _Slist_node_base *__head, const _Slist_node_base *__node)`
- `_Slist_node_base * \_\_slist\_reverse (_Slist_node_base *__node)`
- `size_t \_\_slist\_size (_Slist_node_base *__node)`
- `void \_\_slist\_splice\_after (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_node_base *__before_last)`
- `void \_\_slist\_splice\_after (_Slist_node_base *__pos, _Slist_node_base *__head)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_static\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_static\_pointer\_cast (_FromType *__arg)`

- `size_t __stl_hash_string` (const char \* \_\_s)
- `unsigned long __stl_next_prime` (unsigned long \_\_n)
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`  
`_Ret __stoa` (\_TRet(\*\_\_convf)(const \_CharT \*, \_CharT \*\*, \_Base...), const char  
\* \_\_name, const \_CharT \* \_\_str, std::size\_t \* \_\_idx, \_Base... \_\_base)
- `void __throw_concurrency_lock_error` ()
- `void __throw_concurrency_unlock_error` ()
- `void __throw_forced_error` ()
- `template<typename _String, typename _CharT = typename _String::value_type>`  
`_String __to_xstring` (int(\*\_\_convf)(\_CharT \*, std::size\_t, const \_CharT \*, \_\_-  
builtin\_va\_list), std::size\_t \_\_n, const \_CharT \* \_\_fmt,...)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
[pair](#)< \_InputIter, \_ForwardIter > `__uninitialized_copy_n` (\_InputIter \_\_first, \_-  
Size \_\_count, \_ForwardIter \_\_result, [std::input\\_iterator\\_tag](#))
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`  
[pair](#)< \_RandomAccessIter, \_ForwardIter > `__uninitialized_copy_n` (\_-  
RandomAccessIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::random\\_-](#)  
[access\\_iterator\\_tag](#))
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
[pair](#)< \_InputIter, \_ForwardIter > `__uninitialized_copy_n` (\_InputIter \_\_first, \_-  
Size \_\_count, \_ForwardIter \_\_result)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
[pair](#)< \_InputIter, \_ForwardIter > `__uninitialized_copy_n_a` (\_InputIter \_\_first,  
\_Size \_\_count, \_ForwardIter \_\_result, \_Allocator \_\_alloc)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`  
[pair](#)< \_InputIter, \_ForwardIter > `__uninitialized_copy_n_a` (\_InputIter \_\_first,  
\_Size \_\_count, \_ForwardIter \_\_result, [std::allocator](#)< \_Tp >)
- `void __verbose_terminate_handler` ()
- `size_t __Bit_scan_forward` (size\_t \_\_num)
- `template<typename _ForwardIterator, typename _Allocator >`  
`void __Destroy_const` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_-  
Allocator \_\_alloc)
- `template<typename _ForwardIterator, typename _Tp >`  
`void __Destroy_const` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [alloca-](#)  
[tor](#)< \_Tp >)
- `template<class _CharT, class _Traits >`  
`void __Rope_fill` ([basic\\_ostream](#)< \_CharT, \_Traits > & \_\_o, size\_t \_\_n)
- `template<class _CharT >`  
`bool __Rope_is_simple` (\_CharT \*)
- `bool __Rope_is_simple` (char \*)
- `bool __Rope_is_simple` (wchar\_t \*)
- `template<class _Rope_iterator >`  
`void __Rope_rotate` (\_Rope\_iterator \_\_first, \_Rope\_iterator \_\_middle, \_Rope \_-  
iterator \_\_last)

- `template<class _CharT >`  
`void _S_cond_store_eos (_CharT &)`
- `void _S_cond_store_eos (char &__c)`
- `void _S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`  
`_CharT _S_eos (_CharT *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`  
`bool _S_is_basic_char_type (_CharT *)`
- `bool _S_is_basic_char_type (char *)`
- `bool _S_is_one_byte_char_type (char *)`
- `template<class _CharT >`  
`bool _S_is_one_byte_char_type (_CharT *)`
- `template<class _Operation1 , class _Operation2 >`  
`unary\_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`  
`binary\_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`  
`constant\_void\_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant\_unary\_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant\_binary\_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator , typename _Size , typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > copy\_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator , typename _Tp , typename _Size >`  
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator , typename _Predicate , typename _Size >`  
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator , typename _Distance >`  
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`  
`_Tp identity\_element (std::plus< _Tp >)`
- `template<class _Tp >`  
`_Tp identity\_element (std::multiplies< _Tp >)`
- `static _Atomic_word int __val if (__gthread_active_p()) return __exchange_and_add(__mem`



- `template<typename _ForwardIter, typename _Tp >`  
`void iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`  
`bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`  
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`  
`bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _-`  
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`  
`bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int lexicographical\_compare\_3way (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem\_fun1\_t< _Ret, _Tp, _Arg > mem\_fun1 (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem\_fun1\_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator!= (const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &_  
__hs1, const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs,`  
`const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs,`  
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp, typename _Array >`  
`bool operator!= (const array\_allocator< _Tp, _Array > &, const array\_-`  
`allocator< _Tp, _Array > &)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator<`  
`_Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`  
`typename > class _Base>`  
`bool operator!= (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits,`  
`_Alloc, _Base > &__rhs)`
- `template<typename _Tp >`  
`bool operator!= (const malloc\_allocator< _Tp > &, const malloc\_allocator<`  
`_Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const \_Rope\_const\_iterator< _CharT, _Alloc > &__x, const`  
`\_Rope\_const\_iterator< _CharT, _Alloc > &__y)`

- `template<typename _Tp, typename _Poolp >`  
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`  
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp >`  
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`  
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp >`  
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Tp, typename _Cond >`  
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Tp, class _Alloc >`  
`bool operator!= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Cond >`  
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- `__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _Cond >`  
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const _CharT * __right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`auto operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)-> decltype(__lhs.base()-__rhs.base())`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container >::difference_type operator- (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT,`  
`_Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _-`  
`Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Cond >`  
`throw_value_base< _Cond > operator- (const throw_value_base< _Cond >`  
`&__a, const throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`  
`bool operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc`  
`> &_SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator< (const __normal_iterator< _IteratorL, _Container > &__lhs,`  
`const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator< (const __normal_iterator< _Iterator, _Container > &__lhs,`  
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const`  
`_Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _-`  
`Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename V, typename I, typename S >`  
`bool operator< (const character< V, I, S > &lhs, const character< V, I, S >`  
`&rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_-`  
`adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT,`  
`_Alloc > &__right)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`  
`typename > class _Base>`  
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits,`  
`_Alloc, _Base > &__rhs)`
- `template<typename _Cond >`  
`bool operator< (const throw_value_base< _Cond > &__a, const throw_-`  
`value_base< _Cond > &__b)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`  
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_rhs)`
- `template<class _Tp, class _Alloc >`  
`bool operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const _CharT * __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator== (const __normal_iterator< _IteratorL, _Container > &_lhs, const __normal_iterator< _IteratorR, _Container > &_rhs)`
- `template<typename _Tp >`  
`bool operator== (const _Pointer_adapter< _Tp > &_lhs, int __rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator== (const __normal_iterator< _Iterator, _Container > &_lhs, const __normal_iterator< _Iterator, _Container > &_rhs)`
- `template<typename _Tp >`  
`bool operator== (int __lhs, const _Pointer_adapter< _Tp > &_rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &_hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &_hm2)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &_x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &_y)`
- `template<typename _Tp, typename _Cond >`  
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<typename _Tp >`  
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const rope< _CharT, _Alloc > &_left, const rope< _CharT, _Alloc > &_right)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &_ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &_ht2)`

- `template<class _Tp, class _Alloc >`  
`bool operator== (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`  
`__enable_if< std::__is_char< _CharT >::__value, bool >::__type operator== (const \_\_versa\_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const \_\_versa\_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const \_Rope\_iterator< _CharT, _Alloc > &__x, const \_Rope\_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool operator== (const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp >`  
`bool operator== (const malloc\_allocator< _Tp > &, const malloc\_allocator< _Tp > &)`
- `template<typename _Tp, typename _Poolp >`  
`bool operator== (const \_\_mt\_alloc< _Tp, _Poolp > &, const \_\_mt\_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`  
`bool operator== (const new\_allocator< _Tp > &, const new\_allocator< _Tp > &)`
- `template<typename V, typename I, typename S >`  
`bool operator== (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator== (const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (_Tp1 __lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp >`  
`bool operator== (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`



- bool **operator==** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &\_  
\_lhs, const `_CharT` \*\_\_rhs)
- template<typename `_Cond` >  
bool **operator==** (const `throw_value_base`< `_Cond` > &\_\_a, const `throw_`-  
`value_base`< `_Cond` > &\_\_b)
- template<typename `_Tp`, typename `_Array` >  
bool **operator==** (const `array_allocator`< `_Tp`, `_Array` > &, const `array_`-  
`allocator`< `_Tp`, `_Array` > &)
- template<class `_CharT`, class `_Alloc` >  
bool **operator==** (const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &\_\_x, const  
`_Rope_const_iterator`< `_CharT`, `_Alloc` > &\_\_y)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename,  
typename > class `_Base`>  
bool **operator==** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &\_  
\_lhs, const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &\_\_rhs)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename,  
typename > class `_Base`>  
bool **operator==** (const `_CharT` \*\_\_lhs, const `__versa_string`< `_CharT`, `_Traits`,  
`_Alloc`, `_Base` > &\_\_rhs)
- template<typename `_IteratorL`, typename `_IteratorR`, typename `_Container` >  
bool **operator>** (const `__normal_iterator`< `_IteratorL`, `_Container` > &\_\_lhs,  
const `__normal_iterator`< `_IteratorR`, `_Container` > &\_\_rhs)
- template<typename `_Tp` >  
bool **operator>** (const `_Pointer_adapter`< `_Tp` > &\_\_lhs, const `_Pointer_`-  
`adapter`< `_Tp` > &\_\_rhs)
- template<typename `_Iterator`, typename `_Container` >  
bool **operator>** (const `__normal_iterator`< `_Iterator`, `_Container` > &\_\_lhs,  
const `__normal_iterator`< `_Iterator`, `_Container` > &\_\_rhs)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename,  
typename > class `_Base`>  
bool **operator>** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &\_  
lhs, const `_CharT` \*\_\_rhs)
- template<class `_CharT`, class `_Alloc` >  
bool **operator>** (const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &\_\_x, const  
`_Rope_const_iterator`< `_CharT`, `_Alloc` > &\_\_y)
- template<typename `_Tp1`, typename `_Tp2` >  
bool **operator>** (const `_Pointer_adapter`< `_Tp1` > &\_\_lhs, `_Tp2` \_\_rhs)
- template<class `_CharT`, class `_Alloc` >  
bool **operator>** (const `_Rope_iterator`< `_CharT`, `_Alloc` > &\_\_x, const `_`-  
`Rope_iterator`< `_CharT`, `_Alloc` > &\_\_y)
- template<typename `_Tp1`, typename `_Tp2` >  
bool **operator>** (`_Tp1` \_\_lhs, const `_Pointer_adapter`< `_Tp2` > &\_\_rhs)
- template<class `_CharT`, class `_Alloc` >  
bool **operator>** (const `rope`< `_CharT`, `_Alloc` > &\_\_x, const `rope`< `_CharT`,  
`_Alloc` > &\_\_y)

- `template<typename _Tp1, typename _Tp2 >`  
`bool operator> (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<class _Tp, class _Alloc >`  
`bool operator> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const \_Rope\_const\_iterator< _CharT, _Alloc > &__x, const \_Rope\_const\_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator>= (const \_\_normal\_iterator< _Iterator, _Container > &__lhs, const \_\_normal\_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp >`  
`bool operator>= (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const \_Rope\_iterator< _CharT, _Alloc > &__x, const \_Rope\_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (const \_Pointer\_adapter< _Tp1 > &__lhs, \_Tp2 __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator>= (const \_\_normal\_iterator< _IteratorL, _Container > &__lhs, const \_\_normal\_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- bool **operator**>= (const `__versa_string`< \_CharT, \_Traits, \_Alloc, \_Base > &\_  
\_lhs, const \_CharT \* \_\_rhs)
- template<typename \_Tp1, typename \_Tp2 >  
bool **operator**>= (\_Tp1 \_\_lhs, const `__Pointer_adapter`< \_Tp2 > & \_\_rhs)
- template<class \_Tp, class \_Alloc >  
bool **operator**>= (const `slist`< \_Tp, \_Alloc > &\_SL1, const `slist`< \_Tp, \_Alloc  
> &\_SL2)
- template<typename \_Tp1, typename \_Tp2 >  
bool **operator**>= (const `__Pointer_adapter`< \_Tp1 > & \_\_lhs, const `__Pointer_ -`  
`adapter`< \_Tp2 > & \_\_rhs)
- template<typename \_Tp, typename \_Integer, typename \_MonoidOperation >  
\_Tp **power** (\_Tp \_\_x, \_Integer \_\_n, \_MonoidOperation \_\_monoid\_op)
- template<typename \_Tp, typename \_Integer >  
\_Tp **power** (\_Tp \_\_x, \_Integer \_\_n)
- template<typename \_InputIterator, typename \_RandomAccessIterator, typename \_  
RandomNumberGenerator >  
\_RandomAccessIterator **random\_sample** (\_InputIterator \_\_first, \_InputIterator  
\_\_last, \_RandomAccessIterator \_\_out\_first, \_RandomAccessIterator \_\_out\_last,  
\_RandomNumberGenerator & \_\_rand)
- template<typename \_InputIterator, typename \_RandomAccessIterator >  
\_RandomAccessIterator **random\_sample** (\_InputIterator \_\_first, \_InputIterator  
\_\_last, \_RandomAccessIterator \_\_out\_first, \_RandomAccessIterator \_\_out\_last)
- template<typename \_ForwardIterator, typename \_OutputIterator, typename \_Distance, typename  
\_RandomNumberGenerator >  
\_OutputIterator **random\_sample\_n** (\_ForwardIterator \_\_first, \_ForwardIterator  
\_\_last, \_OutputIterator \_\_out, const \_Distance \_\_n, \_RandomNumberGenerator  
& \_\_rand)
- template<typename \_ForwardIterator, typename \_OutputIterator, typename \_Distance >  
\_OutputIterator **random\_sample\_n** (\_ForwardIterator \_\_first, \_ForwardIterator  
\_\_last, \_OutputIterator \_\_out, const \_Distance \_\_n)
- void **rotate** (\_Rope\_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> \_\_first, \_Rope\_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> \_\_  
middle, \_Rope\_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> \_\_  
last)
- double **stod** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0)
- float **stof** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0)
- int **stoi** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0, int \_\_base=10)
- long **stol** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0, int \_\_base=10)
- long double **stold** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0)
- long long **stoll** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0, int \_\_base=10)
- unsigned long **stoul** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx=0, int \_\_  
base=10)
- unsigned long long **stoull** (const `__vstring` & \_\_str, std::size\_t \* \_\_idx, int \_\_  
base=10)

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash\_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash\_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _CharT, class __Alloc >`  
`void swap (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)`
- `template<typename _Cond >`  
`void swap (throw\_value\_base< _Cond > &__a, throw\_value\_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`  
`void swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`  
`void swap (_ExtPtr_allocator< _Tp > &__larg, ExtPtr\_allocator< _Tp > &__rarg)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash\_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash\_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`  
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`
- `template<class _CharT, class _Alloc >`  
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`
- `\_\_vstring to\_string (long double __val)`
- `\_\_vstring to\_string (double __val)`
- `\_\_vstring to\_string (float __val)`
- `\_\_vstring to\_string (unsigned __val)`
- `\_\_vstring to\_string (unsigned long __val)`
- `\_\_vstring to\_string (unsigned long long __val)`
- `\_\_vstring to\_string (int __val)`
- `\_\_vstring to\_string (long long __val)`
- `\_\_vstring to\_string (long __val)`
- `\_\_wvstring to\_wstring (double __val)`
- `\_\_wvstring to\_wstring (long long __val)`
- `\_\_wvstring to\_wstring (long double __val)`

- [\\_\\_wvstring to\\_wstring](#) (long \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (float \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (unsigned long \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (unsigned long long \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (unsigned \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (int \_\_val)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
[pair< \\_InputIter, \\_ForwardIter > uninitialized\\_copy\\_n](#) (`_InputIter __first, _Size __count, _ForwardIter __result`)

## Variables

- static const `_Lock_policy` `__default_lock_policy`
- static const unsigned long `__stl_prime_list` [`_S_num_primes`]
- static `_Atomic_word` int `__val` `__val`
- [rope](#)< `_CharT, _Alloc` > `identity_element` (`_Rope_Concat_fn`< `_CharT, _Alloc` >)

### 4.1.1 Detailed Description

GNU extensions for public use.

### 4.1.2 Function Documentation

**4.1.2.1** `template<typename _ToType, typename _FromType > _ToType`  
`__gnu_cxx::__static_pointer_cast ( const _FromType & __arg )`  
**[inline]**

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 61 of file `cast.h`.

References `__static_pointer_cast()`.

Referenced by `__static_pointer_cast()`.

**4.1.2.2** `template<typename _ToType, typename _FromType > _ToType`  
`__gnu_cxx::__static_pointer_cast ( _FromType * __arg )` **[inline]**

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 89 of file cast.h.

References `__static_pointer_cast()`.

#### 4.1.2.3 `size_t __gnu_cxx::Bit_scan_forward ( size_t __num ) [inline]`

Generic Version of the bsf instruction.

Definition at line 512 of file bitmap\_allocator.h.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

#### 4.1.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test difference of C string and string.

##### Parameters

`__lhs` C string.

`__rhs` String.

##### Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2175 of file vstring.h.

#### 4.1.2.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test difference of two strings.

##### Parameters

`__lhs` First string.

`__rhs` Second string.

##### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2162 of file vstring.h.

**4.1.2.6** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator!=( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test difference of string and C string.

#### Parameters

`__lhs` String.

`__rhs` C string.

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2188 of file `vstring.h`.

**4.1.2.7** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >  
__gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __lhs, _CharT __rhs )`

Concatenate string and character.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 239 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.8** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >  
__gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __rhs )`

Concatenate two strings.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 179 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.9** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >  
__gnu_cxx::operator+ ( const _CharT * __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate C string and string.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 192 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.



**4.1.2.10** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >  
__gnu_cxx::operator+ ( _CharT __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate character and string.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 209 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.11** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >  
__gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT * __rhs )`

Concatenate string and C string.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 222 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.12** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator< ( const _CharT * __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string precedes string.

#### Parameters

`__lhs` C string.

`__rhs` String.

#### Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2228 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.13** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator< ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string precedes C string.

#### Parameters

`__lhs` String.

`__rhs` C string.

#### Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2215 of file `vstring.h`.

**4.1.2.14** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator< ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string precedes string.

**Parameters**

`__lhs` First string.  
`__rhs` Second string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2202 of file `vstring.h`.

```
4.1.2.15 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator<= (const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
[inline]
```

Test if C string doesn't follow string.

**Parameters**

`__lhs` C string.  
`__rhs` String.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2308 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.16 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator<= (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't follow string.

**Parameters**

`__lhs` First string.  
`__rhs` Second string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2282 of file `vstring.h`.

```
4.1.2.17 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator<= (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT* __rhs) [inline]
```

Test if string doesn't follow C string.

**Parameters**

`__lhs` String.  
`__rhs` C string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2295 of file `vstring.h`.

```
4.1.2.18 template<typename _Tp > bool __gnu_cxx::operator== (const
_Pointer_adapter< _Tp > & __lhs, const _Pointer_adapter< _Tp >
& __rhs) [inline]
```

Comparison operators for [\\_Pointer\\_adapter](#) defer to the base class's comparison operators, when possible.

Definition at line 529 of file `pointer.h`.

```
4.1.2.19 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator== (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT* __rhs) [inline]
```

Test equivalence of string and C string.

**Parameters**

`__lhs` String.  
`__rhs` C string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2148 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.20 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator==(const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs) [inline]
```

Test equivalence of two strings.

**Parameters**

`__lhs` First string.

`__rhs` Second string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2111 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.21 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator==(const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
[inline]
```

Test equivalence of C string and string.

**Parameters**

`__lhs` C string.

`__rhs` String.

**Returns**

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2135 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.22 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator> (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs) [inline]
```

Test if string follows C string.

#### Parameters

`__lhs` String.  
`__rhs` C string.

#### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2255 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.23 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator> (const _CharT * __lhs, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string follows string.

#### Parameters

`__lhs` C string.  
`__rhs` String.

#### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2268 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.24** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator> ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test if string follows string.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

#### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2242 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.25** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't precede string.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

#### Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2322 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.26** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base>  
bool __gnu_cxx::operator>= ( const _CharT * __lhs, const  
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )  
[inline]`

Test if C string doesn't precede string.

#### Parameters

`__lhs` C string.

`__rhs` String.

#### Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2348 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.27** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't precede C string.

#### Parameters

`__lhs` String.

`__rhs` C string.

#### Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2335 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.



**4.1.2.28** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Swap contents of two strings.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2362 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap()`.

## 4.2 `__gnu_cxx::__detail` Namespace Reference

Implementation details not part of the namespace `__gnu_cxx` interface.

### Classes

- class `__mini_vector`  
*`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.*
- class `_Bitmap_counter`  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class `_Ffit_finder`  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*

### Enumerations

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

## Functions

- void [\\_\\_bit\\_allocate](#) (size\_t \*\_\_pmap, size\_t \_\_pos) throw ()
- void [\\_\\_bit\\_free](#) (size\_t \*\_\_pmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator [\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
size\_t [\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
size\_t [\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)

### 4.2.1 Detailed Description

Implementation details not part of the namespace [\\_\\_gnu\\_cxx](#) interface.

### 4.2.2 Function Documentation

**4.2.2.1 void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) ( size\_t \* [\\_\\_pmap](#), size\_t [\\_\\_pos](#) ) throw () [inline]**

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 491 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

**4.2.2.2 void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) ( size\_t \* [\\_\\_pmap](#), size\_t [\\_\\_pos](#) ) throw () [inline]**

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 502 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`.

**4.2.2.3 template<typename \_AddrPair > size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps](#) ( \_AddrPair \_\_ap ) [inline]**

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 279 of file `bitmap_allocator.h`.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::M_allocate_single_object()`,  
and `__gnu_cxx::bitmap_allocator< _Tp >::M_deallocate_single_object()`.

**4.2.2.4** `template<typename _AddrPair > size_t __gnu_cxx::detail::__num_blocks ( _AddrPair __ap )`  
`[inline]`

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 271 of file `bitmap_allocator.h`.

Referenced by `__num_bitmaps()`.

## 4.3 `__gnu_cxx::typelist` Namespace Reference

GNU `typelist` extensions for public compile-time use.

### Functions

- `template<typename Fn , typename Typelist >`  
`void apply (Fn &, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV >`  
`void apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist >`  
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV >`  
`void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn , typename Typelist >`  
`void apply\_generator (Gn &, Typelist)`

### 4.3.1 Detailed Description

GNU `typelist` extensions for public compile-time use.

### 4.3.2 Function Documentation

**4.3.2.1** `template<typename Gn , typename Typelist > void`  
`__gnu_cxx::typelist::apply_generator ( Gn &, Typelist )`

Apply all `typelist` types to generator functor.

## 4.4 `__gnu_debug` Namespace Reference

GNU debug classes for public use.

### Classes

- struct [\\_\\_is\\_same](#)
- class [\\_\\_After\\_nth\\_from](#)
- class [\\_\\_Not\\_equal\\_to](#)
- class [\\_\\_Safe\\_iterator](#)  
*Safe iterator wrapper.*
- class [\\_\\_Safe\\_iterator\\_base](#)  
*Basic functionality for a safe iterator.*
- class [\\_\\_Safe\\_sequence](#)  
*Base class for constructing a safe sequence type that tracks iterators that reference it.*
- class [\\_\\_Safe\\_sequence\\_base](#)  
*Base class that supports tracking of iterators that reference a sequence.*
- class [basic\\_string](#)  
*Class `std::basic_string` with safety/checking/debug instrumentation.*

### Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

### Enumerations

- enum [\\_Debug\\_msg\\_id](#) {  
[\\_\\_msg\\_valid\\_range](#), [\\_\\_msg\\_insert\\_singular](#), [\\_\\_msg\\_insert\\_different](#), [\\_\\_msg\\_erase\\_bad](#),  
[\\_\\_msg\\_erase\\_different](#), [\\_\\_msg\\_subscript\\_oob](#), [\\_\\_msg\\_empty](#), [\\_\\_msg\\_unpartitioned](#),  
[\\_\\_msg\\_unpartitioned\\_pred](#), [\\_\\_msg\\_unsorted](#), [\\_\\_msg\\_unsorted\\_pred](#), [\\_\\_msg\\_not\\_heap](#),  
[\\_\\_msg\\_not\\_heap\\_pred](#), [\\_\\_msg\\_bad\\_bitset\\_write](#), [\\_\\_msg\\_bad\\_bitset\\_read](#),  
[\\_\\_msg\\_bad\\_bitset\\_flip](#),

```

__msg_self_splice, __msg_splice_alloc, __msg_splice_bad, __msg_splice_
other,
__msg_splice_overlap, __msg_init_singular, __msg_init_copy_singular, __
msg_init_const_singular,
__msg_copy_singular, __msg_bad_deref, __msg_bad_inc, __msg_bad_dec,
__msg_iter_subscript_oob, __msg_advance_oob, __msg_retreat_oob, __
msg_iter_compare_bad,
__msg_compare_different, __msg_iter_order_bad, __msg_order_different,
__msg_distance_bad,
__msg_distance_different, __msg_deref_istream, __msg_inc_istream, __
msg_output_ostream,
__msg_deref_istreambuf, __msg_inc_istreambuf }

```

## Functions

- template<typename \_Iterator >  
bool [\\_\\_check\\_dereferenceable](#) (\_Iterator &)
- template<typename \_Tp >  
bool [\\_\\_check\\_dereferenceable](#) (const \_Tp \*\_\_ptr)
- template<typename \_Iterator, typename \_Sequence >  
bool [\\_\\_check\\_dereferenceable](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence > &\_\_x)
- template<typename \_ForwardIterator, typename \_Tp >  
bool [\\_\\_check\\_partitioned\\_lower](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_  
\_\_last, const \_Tp &\_\_value)
- template<typename \_ForwardIterator, typename \_Tp, typename \_Pred >  
bool [\\_\\_check\\_partitioned\\_lower](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_  
\_\_last, const \_Tp &\_\_value, \_Pred \_\_pred)
- template<typename \_ForwardIterator, typename \_Tp >  
bool [\\_\\_check\\_partitioned\\_upper](#) (\_ForwardIterator \_\_first, \_ForwardIterator  
\_\_last, const \_Tp &\_\_value)
- template<typename \_ForwardIterator, typename \_Tp, typename \_Pred >  
bool [\\_\\_check\\_partitioned\\_upper](#) (\_ForwardIterator \_\_first, \_ForwardIterator  
\_\_last, const \_Tp &\_\_value, \_Pred \_\_pred)
- template<typename \_Iterator >  
bool [\\_\\_check\\_singular](#) (\_Iterator &\_\_x)
- template<typename \_Iterator, typename \_Sequence >  
bool [\\_\\_check\\_singular](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence > &\_\_x)
- template<typename \_Tp >  
bool [\\_\\_check\\_singular](#) (const \_Tp \*\_\_ptr)
- bool [\\_\\_check\\_singular\\_aux](#) (const void \*)
- bool [\\_\\_check\\_singular\\_aux](#) (const [\\_Safe\\_iterator\\_base](#) \* \_\_x)

- `template<typename _InputIterator >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::\_\_true\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::\_\_false\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::\_\_false\_type)`
- `template<typename _CharT >`  
`const _CharT * \_\_check\_string (const _CharT *__s)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * \_\_check\_string (const _CharT *__s, const _Integer &__n __attribute__((__unused__)))`
- `template<typename _InputIterator >`  
`_InputIterator __check_valid_range (const _InputIterator &__first, const _InputIterator &__last __attribute__((__unused__)))`

- `template<typename _InputIterator >`  
`bool __valid_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first,`  
`const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _Integral >`  
`bool __valid_range_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`  
`bool __valid_range_aux (const _InputIterator &__first, const _InputIterator &__-`  
`__last, std::__false_type)`
- `template<typename _InputIterator >`  
`bool __valid_range_aux2 (const _InputIterator &, const _InputIterator &,`  
`std::input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`bool __valid_range_aux2 (const _RandomAccessIterator &__first, const _-`  
`RandomAccessIterator &__last, std::random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > &getline (std::basic_istream< _CharT,`  
`_Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT`  
`__delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > &getline (std::basic_istream< _CharT,`  
`_Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _-`  
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`_Safe_iterator< _Iterator, _Sequence > operator+ (typename _Safe_iterator<`  
`_Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _`  
`_Sequence > &__i)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const`  
`basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`_Safe_iterator< _Iterator, _Sequence >::difference_type operator- (const _`  
`_Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator,`  
`_Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`_Safe_iterator< _IteratorL, _Sequence >::difference_type operator- (const _`  
`_Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _`  
`IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`  
`_CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator >`  
`&__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`



- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string<`  
`_CharT, _Traits, _Allocator > &__rhs)`

### 4.4.1 Detailed Description

GNU debug classes for public use.

### 4.4.2 Function Documentation

#### 4.4.2.1 `template<typename _Iterator > bool __gnu_debug::__check_dereferenceable ( _Iterator & )` `[inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 69 of file functions.h.

#### 4.4.2.2 `template<typename _Tp > bool __gnu_debug::__check_dereferenceable ( const _Tp * __ptr )` `[inline]`

Non-NULL pointers are dereferenceable.

Definition at line 75 of file functions.h.

**4.4.2.3** `template<typename _Iterator, typename _Sequence > bool  
__gnu_debug::__check_dereferenceable ( const _Safe_iterator<  
_Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 81 of file `functions.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**4.4.2.4** `template<typename _Iterator, typename _Sequence > bool  
__gnu_debug::__check_singular ( const _Safe_iterator< _Iterator,  
_Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 62 of file `functions.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.5** `template<typename _Tp > bool __gnu_debug::__check_singular (   
const _Tp * __ptr ) [inline]`

Non-NULL pointers are nonsingular.

Definition at line 56 of file `functions.h`.

**4.4.2.6** `bool __gnu_debug::__check_singular_aux ( const _Safe_iterator_base *  
__x ) [inline]`

Iterators that derive from [\\_Safe\\_iterator\\_base](#) but that aren't `_Safe_iterators` can be determined singular or non-singular via [\\_Safe\\_iterator\\_base](#).

Definition at line 48 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.7** `template<typename _CharT > const _CharT*  
__gnu_debug::__check_string ( const _CharT * __s ) [inline]`

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 175 of file `functions.h`.

**4.4.2.8** `template<typename _CharT, typename _Integer > const _CharT*  
__gnu_debug::__check_string ( const _CharT * __s, const _Integer  
&__n __attribute__((unused)) ) [inline]`

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 163 of file `functions.h`.

**4.4.2.9** `template<typename _InputIterator > bool __gnu_debug::__valid_  
range ( const _InputIterator & __first, const _InputIterator & __last )  
[inline]`

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 133 of file `functions.h`.

References `__valid_range_aux()`.

**4.4.2.10** `template<typename _Iterator, typename _Sequence > bool  
__gnu_debug::__valid_range ( const _Safe_iterator< _Iterator,  
_Sequence > & __first, const _Safe_iterator< _Iterator, _Sequence >  
& __last ) [inline]`

Safe iterators know how to check if they form a valid range.

Definition at line 142 of file `functions.h`.

**4.4.2.11** `template<typename _Integral > bool __gnu_debug::__valid_range_  
aux ( const _Integral &, const _Integral &, std::__true_type )  
[inline]`

We say that integral types form a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 110 of file `functions.h`.

Referenced by `__valid_range()`.

**4.4.2.12** `template<typename _InputIterator > bool __gnu_debug::__valid_  
range_aux ( const _InputIterator & __first, const _InputIterator &  
__last, std::__false_type ) [inline]`

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 118 of file `functions.h`.

References `__valid_range_aux2()`.

**4.4.2.13** `template<typename _InputIterator > bool __gnu_debug::__valid_range_aux2 ( const _InputIterator &, const _InputIterator &, std::input_iterator_tag ) [inline]`

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 100 of file `functions.h`.

**4.4.2.14** `template<typename _RandomAccessIterator > bool __gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator & __first, const _RandomAccessIterator & __last, std::random_access_iterator_tag ) [inline]`

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 89 of file `functions.h`.

Referenced by `__valid_range_aux()`.

## 4.5 `__gnu_internal` Namespace Reference

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

### 4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 4.6 `__gnu_parallel` Namespace Reference

GNU parallel code for public use.

### Classes

- struct [\\_\\_accumulate\\_binop\\_reduct](#)

*General reduction, using a binary operator.*

- struct [\\_\\_accumulate\\_selector](#)  
*[std::accumulate\(\)](#) selector.*
- struct [\\_\\_adjacent\\_difference\\_selector](#)  
*Selector that returns the difference between two adjacent [\\_\\_elements](#).*
- struct [\\_\\_adjacent\\_find\\_selector](#)  
*Test predicate on two adjacent elements.*
- class [\\_\\_binder1st](#)  
*Similar to [std::binder1st](#), but giving the argument types explicitly.*
- class [\\_\\_binder2nd](#)  
*Similar to [std::binder2nd](#), but giving the argument types explicitly.*
- struct [\\_\\_count\\_if\\_selector](#)  
*[std::count\\_if\(\)](#) selector.*
- struct [\\_\\_count\\_selector](#)  
*[std::count\(\)](#) selector.*
- struct [\\_\\_fill\\_selector](#)  
*[std::fill\(\)](#) selector.*
- struct [\\_\\_find\\_first\\_of\\_selector](#)  
*Test predicate on several elements.*
- struct [\\_\\_find\\_if\\_selector](#)  
*Test predicate on a single element, used for [std::find\(\)](#) and [std::find\\_if\(\)](#).*
- struct [\\_\\_for\\_each\\_selector](#)  
*[std::for\\_each\(\)](#) selector.*
- struct [\\_\\_generate\\_selector](#)  
*[std::generate\(\)](#) selector.*
- struct [\\_\\_generic\\_find\\_selector](#)  
*Base class of all [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) selectors.*
- struct [\\_\\_generic\\_for\\_each\\_selector](#)

Generic `__selector` for embarrassingly parallel functions.

- struct `__identity_selector`  
*Selector that just returns the passed iterator.*
- struct `__inner_product_selector`  
*`std::inner_product()` selector.*
- struct `__max_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__min_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__mismatch_selector`  
*Test inverted predicate on a single element.*
- struct `__multiway_merge_3_variant_sentinel_switch`  
*Switch for 3-way merging with `__sentinels` turned off.*
- struct `__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with `__sentinels` turned on.*
- struct `__multiway_merge_4_variant_sentinel_switch`  
*Switch for 4-way merging with `__sentinels` turned off.*
- struct `__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 4-way merging with `__sentinels` turned on.*
- struct `__multiway_merge_k_variant_sentinel_switch`  
*Switch for k-way merging with `__sentinels` turned on.*
- struct `__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for k-way merging with `__sentinels` turned off.*
- struct `__replace_if_selector`  
*`std::replace()` selector.*
- struct `__replace_selector`  
*`std::replace()` selector.*

- struct [\\_\\_transform1\\_selector](#)  
*std::transform() \_\_selector, one input sequence variant.*
- struct [\\_\\_transform2\\_selector](#)  
*std::transform() \_\_selector, two input sequences variant.*
- class [\\_\\_unary\\_negate](#)  
*Similar to [std::unary\\_negate](#), but giving the argument types explicitly.*
- struct [\\_DRandomShufflingGlobalData](#)  
*Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_DRSSorterPU](#)  
*Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_DummyReduct](#)  
*Reduction function doing nothing.*
- class [\\_EqualFromLess](#)  
*Constructs predicate for equality from strict weak ordering predicate.*
- struct [\\_EqualTo](#)  
*Similar to [std::equal\\_to](#), but allows two different types.*
- class [\\_GuardedIterator](#)  
*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*
- class [\\_IteratorPair](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_IteratorTriple](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*
- struct [\\_Job](#)  
*One \_\_job for a certain thread.*
- struct [\\_Less](#)  
*Similar to [std::less](#), but allows two different types.*



- class [`\_Lexicographic`](#)  
*Compare \_\_a pair of types lexicographically, ascending.*
- class [`\_LexicographicReverse`](#)  
*Compare \_\_a pair of types lexicographically, descending.*
- class [`\_LoserTree`](#)  
*Stable `_LoserTree` variant.*
- class [`\_LoserTree< false, \_Tp, \_Compare >`](#)  
*Unstable `_LoserTree` variant.*
- class [`\_LoserTreeBase`](#)  
*Guarded loser/tournament tree.*
- class [`\_LoserTreePointer`](#)  
*Stable `_LoserTree` implementation.*
- class [`\_LoserTreePointer< false, \_Tp, \_Compare >`](#)  
*Unstable `_LoserTree` implementation.*
- class [`\_LoserTreePointerBase`](#)  
*Base class of `_Loser` Tree implementation using pointers.*
- class [`\_LoserTreePointerUnguarded`](#)  
*Stable unguarded `_LoserTree` variant storing pointers.*
- class [`\_LoserTreePointerUnguarded< false, \_Tp, \_Compare >`](#)  
*Unstable unguarded `_LoserTree` variant storing pointers.*
- class [`\_LoserTreePointerUnguardedBase`](#)  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- struct [`\_LoserTreeTraits`](#)  
*Traits for determining whether the loser tree should use pointers or copies.*
- class [`\_LoserTreeUnguarded`](#)  
*Stable implementation of unguarded `_LoserTree`.*
- class [`\_LoserTreeUnguarded< false, \_Tp, \_Compare >`](#)  
*Non-Stable implementation of unguarded `_LoserTree`.*

- class [\\_LoserTreeUnguardedBase](#)  
*Base class for unguarded [\\_LoserTree](#) implementation.*
- struct [\\_Multiplies](#)  
*Similar to [std::multiplies](#), but allows two different types.*
- struct [\\_Nothing](#)  
*Functor doing nothing.*
- struct [\\_Piece](#)  
*Subsequence description.*
- struct [\\_Plus](#)  
*Similar to [std::plus](#), but allows two different types.*
- struct [\\_PMWMSSortingData](#)  
*Data accessed by all threads.*
- class [\\_PseudoSequence](#)  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class [\\_PseudoSequenceIterator](#)  
*\_Iterator associated with [\\_\\_gnu\\_parallel::\\_PseudoSequence](#). If features the usual random-access iterator functionality.*
- struct [\\_QSBThreadLocal](#)  
*Information local to one thread in the parallel quicksort run.*
- class [\\_RandomNumber](#)  
*Random number generator, based on the Mersenne twister.*
- class [\\_RestrictedBoundedConcurrentQueue](#)  
*Double-ended queue of bounded size, allowing lock-free atomic access. [push\\_front\(\)](#) and [pop\\_front\(\)](#) must not be called concurrently to each other, while [pop\\_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.*
- struct [\\_SamplingSorter](#)  
*Stable sorting functor.*
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)

*Non-\_\_stable sorting functor.*

- struct [\\_Settings](#)  
*class [\\_Settings](#) /// Run-time settings for the parallel mode including all tunable parameters.*
- struct [\\_SplitConsistently](#)  
*Split consistently.*
- struct [\\_SplitConsistently](#)< false, [\\_RAIter](#), [\\_Compare](#), [\\_SortingPlacesIterator](#) >  
*Split by sampling.*
- struct [\\_SplitConsistently](#)< true, [\\_RAIter](#), [\\_Compare](#), [\\_SortingPlacesIterator](#) >  
*Split by exact splitting.*
- struct [balanced\\_quicksort\\_tag](#)  
*Forces parallel sorting using balanced quicksort at compile time.*
- struct [balanced\\_tag](#)  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct [constant\\_size\\_blocks\\_tag](#)  
*Selects the constant block size variant for `std::find()`.*
- struct [default\\_parallel\\_tag](#)  
*Recommends parallel execution using the default parallel algorithm.*
- struct [equal\\_split\\_tag](#)  
*Selects the equal splitting variant for `std::find()`.*
- struct [exact\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [find\\_tag](#)  
*Base class for `std::find()` variants.*
- struct [growing\\_blocks\\_tag](#)  
*Selects the growing block size variant for `std::find()`.*
- struct [multiway\\_mergesort\\_exact\\_tag](#)  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*

- struct [multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [multiway\\_mergesort\\_tag](#)  
*Forces parallel sorting using multiway mergesort at compile time.*
- struct [omp\\_loop\\_static\\_tag](#)  
*Recommends parallel execution using OpenMP static load-balancing at compile time.*
- struct [omp\\_loop\\_tag](#)  
*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*
- struct [parallel\\_tag](#)  
*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*
- struct [quicksort\\_tag](#)  
*Forces parallel sorting using unbalanced quicksort at compile time.*
- struct [sampling\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [sequential\\_tag](#)  
*Forces sequential execution at compile time.*
- struct [unbalanced\\_tag](#)  
*Recommends parallel execution using static load-balancing at compile time.*

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }

- enum `__FindAlgorithm` { `GROWING_BLOCKS`, `CONSTANT_SIZE_BLOCKS`, `EQUAL_SPLIT` }
- enum `__MultiwayMergeAlgorithm` { `LOSER_TREE` }
- enum `__Parallelism` {  
`sequential`, `parallel_unbalanced`, `parallel_balanced`, `parallel_omp_loop`,  
`parallel_omp_loop_static`, `parallel_taskqueue` }
- enum `__PartialSumAlgorithm` { `RECURSIVE`, `LINEAR` }
- enum `__SortAlgorithm` { `MWMS`, `QS`, `QS_BALANCED` }
- enum `__SplittingAlgorithm` { `SAMPLING`, `EXACT` }

## Functions

- template<typename `_RAIter`, typename `_DifferenceTp` >  
void `__calc_borders` (`_RAIter` `__elements`, `_DifferenceTp` `__length`, `_DifferenceTp` `*__off`)
- template<typename `_Tp` >  
bool `__compare_and_swap` (volatile `_Tp` `*__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- bool `__compare_and_swap_32` (volatile `int32_t` `*__ptr`, `int32_t` `__comparand`, `int32_t` `__replacement`)
- bool `__compare_and_swap_64` (volatile `int64_t` `*__ptr`, `int64_t` `__comparand`, `int64_t` `__replacement`)
- template<typename `_Iter`, typename `_OutputIterator` >  
`_OutputIterator` `__copy_tail` (`std::pair`< `_Iter`, `_Iter` > `__b`, `std::pair`< `_Iter`, `_Iter` > `__e`, `_OutputIterator` `__r`)
- void `__decode2` (`_CASable` `__x`, int &`__a`, int &`__b`)
- template<typename `_RAIter`, typename `_DifferenceTp` >  
void `__determine_samples` (`_PMWMSortingData`< `_RAIter` > `*__sd`, `_DifferenceTp` `__num_samples`)
- `_CASable` `__encode2` (int `__a`, int `__b`)
- template<typename `_Tp` >  
`_Tp` `__fetch_and_add` (volatile `_Tp` `*__ptr`, `_Tp` `__addend`)
- `int32_t` `__fetch_and_add_32` (volatile `int32_t` `*__ptr`, `int32_t` `__addend`)
- `int64_t` `__fetch_and_add_64` (volatile `int64_t` `*__ptr`, `int64_t` `__addend`)
- template<typename `_RAIter1`, typename `_RAIter2`, typename `_Pred`, typename `_Selector` >  
`std::pair`< `_RAIter1`, `_RAIter2` > `__find_template` (`_RAIter1` `__begin1`, `_RAIter1` `__end1`, `_RAIter2` `__begin2`, `_Pred` `__pred`, `_Selector` `__selector`)
- template<typename `_RAIter1`, typename `_RAIter2`, typename `_Pred`, typename `_Selector` >  
`std::pair`< `_RAIter1`, `_RAIter2` > `__find_template` (`_RAIter1` `__begin1`, `_RAIter1` `__end1`, `_RAIter2` `__begin2`, `_Pred` `__pred`, `_Selector` `__selector`, `equal_split_tag`)

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,`  
`growing\_blocks\_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,`  
`constant\_size\_blocks\_tag)`
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, type-`  
`name _Result >`  
`_UserOp \_\_for\_each\_template\_random\_access (_Iter __begin, _Iter __end,`  
`_UserOp __user_op, _Functionality &__functionality, _Red __reduction, _-`  
`Result __reduction_start, _Result &__output, typename std::iterator\_traits< _-`  
`Iter >::difference_type __bound, \_Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __-`  
`end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, type-`  
`name std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_omp\_loop (_RAIter __begin, _-`  
`RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__-`  
`output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_omp\_loop\_static (_RAIter __begin,`  
`_RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__-`  
`output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_workstealing (_RAIter __begin, _-`  
`RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__-`  
`output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `\_ThreadIndex \_\_get\_max\_threads ()`
- `bool \_\_is\_parallel (const \_Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool \_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter \_\_median\_of\_three\_iterators (_RAIter __a, _RAIter __b, _RAIter __c,`  
`_Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _-`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _-`  
`RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp`  
`__max_length, _Compare __comp)`

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng= RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, type-`

```

name std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __-
num_threads, _RandomNumberGenerator &__rng)
• template<typename _RAIter, typename _RandomNumberGenerator >
void __parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _-
RandomNumberGenerator > *__pus)
• template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __parallel_set_difference (_Iter __begin1, _Iter __end1, _-
Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)
• template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __parallel_set_intersection (_Iter __begin1, _Iter __end1, _-
Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)
• template<typename _Iter, typename _OutputIterator, typename _Operation >
_OutputIterator __parallel_set_operation (_Iter __begin1, _Iter __end1, _-
Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)
• template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __parallel_set_symmetric_difference (_Iter __begin1, _Iter
__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __-
comp)
• template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __-
begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
parallel_tag __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
balanced_quicksort_tag __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
multiway_mergesort_sampling_tag __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _-
Parallelism __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
multiway_mergesort_tag __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
multiway_mergesort_exact_tag __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
quicksort_tag __parallelism)
• template<bool __stable, typename _RAIter, typename _Compare >
void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp,
default_parallel_tag __parallelism)

```



- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`__ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare`  
`__comp, __ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_`  
`RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_`  
`traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits<`  
`_RAIter >::difference_type __num_samples, __ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`__ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _`  
`OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _`  
`OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_conquer (_QSBThreadLocal< _RAIter > ** __tls, _RAIter __begin,`  
`_RAIter __end, _Compare __comp, __ThreadIndex __iam, __ThreadIndex __`  
`num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __qsb_divide (_RAIter __`  
`begin, _RAIter __end, _Compare __comp, __ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > ** __tls,`  
`_Compare &__comp, __ThreadIndex __iam, bool __wait)`
- `template<typename _RandomNumberGenerator >`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`  
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp >`  
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`_RAIter1 __search_template (_RAIter1 __begin1, _RAIter1 __end1, _`  
`RAIter2 __begin2, _RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, type-`  
`name _DifferenceTp, typename _Compare >`  
`_RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_`  
`traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type`  
`>::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_sequential\_random\_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rng)`
- `template<typename _Iter >`  
`void \_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two,`  
`size_t &__range_length)`
- `template<typename _Iter >`  
`void \_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__-`  
`count_to_two, size_t &__range_length, const bool __make_twice)`
- `void \_\_yield ()`
- `template<typename _DifferenceType, typename _OutputIterator >`  
`_OutputIterator equally\_split (_DifferenceType __n, \_ThreadIndex __num_-`  
`threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`  
`_DifferenceType equally\_split\_point (_DifferenceType __n, \_ThreadIndex __-`  
`num_threads, \_ThreadIndex __thread_no)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t list\_partition (const _Iter __begin, const _Iter __end, _Iter *__-`  
`starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __-`  
`oversampling=0)`
- `template<typename _Tp >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename`  
`_Compare >`  
`void multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_-`  
`seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __-`  
`comp=std::less< typename std::iterator_traits< typename std::iterator_traits<`  
`_RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _-`  
`RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp`  
`>())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _-`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _-`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, default\_parallel\_tag __tag)`

- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >`  
`void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename`

```

std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_
type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare
__comp)
• template<bool __stable, typename _RAIterIterator, typename _Compare, typename _
DifferenceType >
void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _
RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __
total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _
DifferenceType > > *__pieces)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _
RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,
_Compare __comp, __gnu_parallel::sequential_tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _
RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,
_Compare __comp, parallel_tag __tag=parallel_tag(0))
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _
RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,
_Compare __comp, __gnu_parallel::exact_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _
RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,
_Compare __comp, default_parallel_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _
RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,
_Compare __comp, sampling_tag __tag)
• template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Splitter, typename _Compare >
_RAIter3 parallel_multiway_merge (_RAIterIterator __seqs_begin, _
RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _
DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
• template<bool __stable, bool __exact, typename _RAIter, typename _Compare >
void parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp,
_ThreadIndex __num_threads)
• template<bool __stable, bool __exact, typename _RAIter, typename _Compare >
void parallel_sort_mwms_pu (_PMWMSortingData< _RAIter > *__sd, _
Compare &__comp)

```

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_CASable\\_mask](#)

### 4.6.1 Detailed Description

GNU parallel code for public use.

### 4.6.2 Typedef Documentation

#### 4.6.2.1 typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

#### 4.6.2.2 typedef int64\_t [\\_\\_gnu\\_parallel::\\_CASable](#)

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

#### 4.6.2.3 typedef uint64\_t [\\_\\_gnu\\_parallel::\\_SequenceIndex](#)

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

#### 4.6.2.4 typedef uint16\_t [\\_\\_gnu\\_parallel::\\_ThreadIndex](#)

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 enum \_\_gnu\_parallel::\_AlgorithmStrategy

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

#### 4.6.3.2 enum \_\_gnu\_parallel::\_FindAlgorithm

Find algorithms:

Definition at line 106 of file types.h.

#### 4.6.3.3 enum \_\_gnu\_parallel::\_MultiwayMergeAlgorithm

Merging algorithms:

Definition at line 85 of file types.h.

#### 4.6.3.4 enum \_\_gnu\_parallel::\_Parallelism

Run-time equivalents for the compile-time tags.

##### Enumerator:

*sequential* Not parallel.

*parallel\_unbalanced* Parallel unbalanced (equal-sized chunks).

*parallel\_balanced* Parallel balanced (work-stealing).

*parallel\_omp\_loop* Parallel with OpenMP dynamic load-balancing.

*parallel\_omp\_loop\_static* Parallel with OpenMP static load-balancing.

*parallel\_taskqueue* Parallel with OpenMP taskqueue construct.

Definition at line 44 of file types.h.

#### 4.6.3.5 enum \_\_gnu\_parallel::\_PartialSumAlgorithm

Partial sum algorithms: recursive, linear.

Definition at line 91 of file types.h.

#### 4.6.3.6 enum \_\_gnu\_parallel::\_SortAlgorithm

Sorting algorithms:

Definition at line 76 of file types.h.

#### 4.6.3.7 enum \_\_gnu\_parallel::\_SplittingAlgorithm

Sorting/merging algorithms: sampling, \_\_exact.

Definition at line 98 of file types.h.

### 4.6.4 Function Documentation

#### 4.6.4.1 template<typename \_RAIter, typename \_DifferenceTp> void \_\_gnu\_parallel::\_calc\_borders ( \_RAIter \_\_elements, \_DifferenceTp \_\_length, \_DifferenceTp \* \_\_off )

Precalculate \_\_advances for Knuth-Morris-Pratt algorithm.

##### Parameters

*\_\_elements* Begin iterator of sequence to search for.

*\_\_length* Length of sequence to search for.

*\_\_advances* Returned \_\_offsets.

Definition at line 51 of file search.h.

Referenced by \_\_search\_template().

#### 4.6.4.2 template<typename \_Tp> bool \_\_gnu\_parallel::\_compare\_and\_swap ( volatile \_Tp \* \_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement ) [inline]

Compare \*\_\_ptr and \_\_comparand. If equal, let \*\_\_ptr=\_\_replacement and return true, return false otherwise.

Implementation is heavily platform-dependent.

##### Parameters

*\_\_ptr* Pointer to signed integer.

*\_\_comparand* Compare value.

*\_\_replacement* Replacement value.



Definition at line 337 of file `parallel/compatibility.h`.

References `__compare_and_swap_32()`, and `__compare_and_swap_64()`.

Referenced by `__parallel_partition()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_back()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_front()`.

#### 4.6.4.3 `bool __gnu_parallel::__compare_and_swap_32 ( volatile int32_t * __ptr, int32_t __comparand, int32_t __replacement ) [inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

##### Parameters

`__ptr` Pointer to 32-bit signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 240 of file `parallel/compatibility.h`.

Referenced by `__compare_and_swap()`.

#### 4.6.4.4 `bool __gnu_parallel::__compare_and_swap_64 ( volatile int64_t * __ptr, int64_t __comparand, int64_t __replacement ) [inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

##### Parameters

`__ptr` Pointer to 64-bit signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 283 of file `parallel/compatibility.h`.

Referenced by `__compare_and_swap()`.

#### 4.6.4.5 `void __gnu_parallel::__decode2 ( _CASable __x, int & __a, int & __b ) [inline]`

Decode two integers from one `gnu_parallel::_CASable`.

**Parameters**

- `__x` [\\_\\_gnu\\_parallel::\\_CASable](#) to decode integers from.
- `__a` First integer, to be decoded from the most-significant `_CASable_bits/2` bits of `__x`.
- `__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits of `__x`.

**See also**

[\\_\\_encode2](#)

Definition at line 133 of file `parallel/base.h`.

References `_CASable_bits`, and `_CASable_mask`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

**4.6.4.6** `template<typename _RAIter, typename _DifferenceTp> void  
__gnu_parallel::_determine_samples ( _PMWMSSortingData<  
_RAIter> * __sd, _DifferenceTp __num_samples )`

Select `_M_samples` from a sequence.

**Parameters**

- `__sd` Pointer to algorithm data. `_Result` will be placed in `__sd->_M_samples`.
- `__num_samples` Number of `_M_samples` to select.

Definition at line 97 of file `multiway_mergesort.h`.

References `__gnu_parallel::_PMWMSSortingData< _RAIter>::_M_samples`, `__gnu_parallel::_PMWMSSortingData< _RAIter>::_M_source`, `__gnu_parallel::_PMWMSSortingData< _RAIter>::_M_starts`, and `equally_split()`.

**4.6.4.7** `_CASable __gnu_parallel::_encode2 ( int __a, int __b ) [inline]`

Encode two integers into one `gnu_parallel::_CASable`.

**Parameters**

- `__a` First integer, to be encoded in the most-significant `_CASable_bits/2` bits.
- `__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits.

**Returns**

value encoding `__a` and `__b`.

**See also**

[\\_\\_decode2](#)

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::push_front()`.

#### 4.6.4.8 `template<typename _Tp> _Tp __gnu_parallel::__fetch_and_add ( volatile _Tp * __ptr, _Tp __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

**Parameters**

`__ptr` Pointer to a signed integer.

`__addend` Value to add.

Definition at line 186 of file `parallel/compatibility.h`.

References `__fetch_and_add_32()`, and `__fetch_and_add_64()`.

Referenced by `__parallel_partition()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::push_front()`.

#### 4.6.4.9 `int32_t __gnu_parallel::__fetch_and_add_32 ( volatile int32_t * __ptr, int32_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

**Parameters**

`__ptr` Pointer to a 32-bit signed integer.

`__addend` Value to add.

Definition at line 95 of file parallel/compatibility.h.

Referenced by `__fetch_and_add()`.

**4.6.4.10** `int64_t __gnu_parallel::__fetch_and_add_64 ( volatile int64_t * __ptr,  
int64_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to a 64-bit signed integer.

`__addend` Value to add.

Definition at line 134 of file parallel/compatibility.h.

Referenced by `__fetch_and_add()`.

**4.6.4.11** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector )  
[inline]`

Parallel `std::find`, switch for different algorithms.

#### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence. Must have same length as first sequence.

`__pred` Find predicate.

`__selector` Functionality (e. g. `std::find_if()`, `std::equal()`,...)

#### Returns

Place of finding in both sequences.

Definition at line 60 of file find.h.

References `__gnu_parallel::_Settings::get()`, and `std::make_pair()`.

**4.6.4.12** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,  
equal_split_tag )`

Parallel std::find, equal splitting variant.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence. Second \_\_sequence must have same length as first sequence.
- `__pred` Find predicate.
- `__selector` \_Functionality (e. g. std::find\_if(), std::equal(),...)

#### Returns

Place of finding in both sequences.

Definition at line 97 of file find.h.

References `_GLIBCXX_CALL`, and `equally_split()`.

**4.6.4.13** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,  
growing_blocks_tag )`

Parallel std::find, growing block size variant.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence. Second \_\_sequence must have same length as first sequence.
- `__pred` Find predicate.
- `__selector` \_Functionality (e. g. std::find\_if(), std::equal(),...)

#### Returns

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_scale\\_factor](#)

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.14** `template<typename _RAIter1, typename _RAIter2, typename  
_Pred, typename _Selector> std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::_find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,  
constant_size_blocks_tag )`

Parallel `std::find`, constant block size variant.

**Parameters**

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence. Second `__sequence` must have same length as first sequence.  
`__pred` Find predicate.  
`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

**Returns**

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_block\\_size](#) There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.15** `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp  
__gnu_parallel::_for_each_template_random_access ( _Iter  
__begin, _Iter __end, _UserOp __user_op, _Functionality &  
__functionality, _Red __reduction, _Result __reduction_start, _Result  
& __output, typename std::iterator_traits< _Iter >::difference_type  
__bound, _Parallelism __parallelism_tag )`

Chose the desired algorithm by evaluating `__parallelism_tag`.

#### Parameters

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__user_op` A user-specified functor (comparator, predicate, associative operator,...)
- `__functionality` functor to *process* an element with `__user_op` (depends on desired functionality, e. g. accumulate, `for_each`,...)
- `__reduction` Reduction functor.
- `__reduction_start` Initial value for reduction.
- `__output` Output iterator.
- `__bound` Maximum number of elements processed.
- `__parallelism_tag` Parallelization method

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

**4.6.4.16** `template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::_for_each_template_random_access_ed ( _RAIter  
__begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result  
__base, _Result & __output, typename std::iterator_traits< _RAIter  
>::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

**Parameters**

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, ...)
- `__f` Functor to "process" an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to "add" a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.
- `__output` Pointer to position where final result is written to
- `__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.17** `template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_omp_loop (  
_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

**Parameters**

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, etc.).
- `__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to *add* a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.



`__output` Pointer to position where final result is written to

`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `omp_loop.h`.

Referenced by `__for_each_template_random_access()`.

```
4.6.4.18 template<typename _RAIter, typename _Op, typename _Fu,
typename _Red, typename _Result> _Op __gnu_parallel::__-
for_each_template_random_access_omp_loop_static (_RAIter
__begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result
__base, _Result & __output, typename std::iterator_traits< _RAIter
>::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

### Parameters

`__begin` Begin iterator of element sequence.

`__end` End iterator of element sequence.

`__o` User-supplied functor (comparator, predicate, adding functor, ...).

`__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).

`__r` Functor to *add* a single `__result` to the already processed `__elements` (depends on functionality).

`__base` Base value for reduction.

`__output` Pointer to position where final result is written to

`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

**4.6.4.19** `template<typename _RAIter , typename _Op , typename  
_Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_workstealing (   
_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

#### Parameters

**\_\_begin** Begin iterator of element sequence.  
**\_\_end** End iterator of element sequence.  
**\_\_op** User-supplied functor (comparator, predicate, adding functor, ...).  
**\_\_f** Functor to *process* an element with \_\_op (depends on desired functionality, e. g. for std::for\_each(), ...).  
**\_\_r** Functor to *add* a single \_\_result to the already processed elements (depends on functionality).  
**\_\_base** Base value for reduction.  
**\_\_output** Pointer to position where final result is written to  
**\_\_bound** Maximum number of elements processed (e. g. for std::count\_n()).

#### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file workstealing.h.

References `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Job< _DifferenceTp >::_M_first`, `__gnu_parallel::_Job< _DifferenceTp >::_M_last`, `__gnu_parallel::_Job< _DifferenceTp >::_M_load`, `__gnu_parallel::_Settings::cache_line_size`, `__gnu_parallel::_Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.20** `template<typename _Iter , typename _Compare > bool  
__gnu_parallel::__is_sorted ( _Iter __begin, _Iter __end,  
_Compare __comp )`

Check whether [`__begin`, `__end`) is sorted according to `__comp`.

#### Parameters

**\_\_begin** Begin iterator of sequence.

**\_\_end** End iterator of sequence.

**\_\_comp** Comparator.

### Returns

true if sorted, false otherwise.

Definition at line 51 of file checkers.h.

Referenced by \_\_sequential\_multiway\_merge(), multiway\_merge\_loser\_tree\_sentinel(), and parallel\_multiway\_merge().

**4.6.4.21** `template<typename _RAIter, typename _Compare> _RAIter  
__gnu_parallel::__median_of_three_iterators ( _RAIter __a, _RAIter  
__b, _RAIter __c, _Compare __comp )`

Compute the median of three referenced elements, according to \_\_comp.

### Parameters

**\_\_a** First iterator.

**\_\_b** Second iterator.

**\_\_c** Third iterator.

**\_\_comp** Comparator.

Definition at line 398 of file parallel/base.h.

Referenced by \_\_qsb\_divide().

**4.6.4.22** `template<typename _RAIter1, typename _RAIter2, typename  
_OutputIterator, typename _DifferenceTp, typename _Compare  
> _OutputIterator __gnu_parallel::__merge_advance ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp ) [inline]`

Merge routine being able to merge only the \_\_max\_length smallest elements.

The \_\_begin iterators are advanced accordingly, they might not reach \_\_end, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

### Parameters

**\_\_begin1** Begin iterator of first sequence.

**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence.  
**\_\_end2** End iterator of second sequence.  
**\_\_target** Target begin iterator.  
**\_\_max\_length** Maximum number of elements to merge.  
**\_\_comp** Comparator.

### Returns

Output end iterator.

Definition at line 171 of file merge.h.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

**4.6.4.23** `template<typename _RAIter1, typename _RAIter2, typename  
_OutputIterator, typename _DifferenceTp, typename _Compare >  
_OutputIterator __gnu_parallel::__merge_advance_movc ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

### Parameters

**\_\_begin1** Begin iterator of first sequence.  
**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence.  
**\_\_end2** End iterator of second sequence.  
**\_\_target** Target begin iterator.  
**\_\_max\_length** Maximum number of elements to merge.  
**\_\_comp** Comparator.

### Returns

Output end iterator.

Definition at line 105 of file merge.h.

Referenced by `__merge_advance()`.

```
4.6.4.24 template<typename _RAIter1 , typename _RAIter2 , typename
 _OutputIterator , typename _DifferenceTp , typename _Compare >
 _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1
 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2
 __end2, _OutputIterator __target, _DifferenceTp __max_length,
 _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__target` Target begin iterator.  
`__max_length` Maximum number of elements to merge.  
`__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 57 of file `merge.h`.

```
4.6.4.25 template<typename _RAIter1 , typename _RAIter2 ,
 typename _RAIter3 , typename _Compare > _RAIter3
 __gnu_parallel::__parallel_merge_advance (_RAIter1 & __begin1,
 _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2,
 _RAIter3 __target, typename std::iterator_traits< _RAIter1
 >::difference_type __max_length, _Compare __comp) [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.

**\_\_target** Target begin iterator.  
**\_\_max\_length** Maximum number of elements to merge.  
**\_\_comp** Comparator.

### Returns

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

```
4.6.4.26 template<typename _RAIter1, typename _RAIter3, typename
 _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (
 _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2,
 _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<
 _RAIter1 >::difference_type __max_length, _Compare __comp)
 [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

### Parameters

**\_\_begin1** Begin iterator of first sequence.  
**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence.  
**\_\_end2** End iterator of second sequence.  
**\_\_target** Target begin iterator.  
**\_\_max\_length** Maximum number of elements to merge.  
**\_\_comp** Comparator.

### Returns

Output end iterator.

Definition at line 223 of file merge.h.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**4.6.4.27** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_nth_element ( _RAIter __begin, _RAIter  
__nth, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::nth_element()`.

#### Parameters

- `__begin` Begin iterator of input sequence.
- `__nth` Iterator of element that must be in position afterwards.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 332 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, `__gnu_parallel::_Settings::partition_minimal_n`, and `std::swap()`.

Referenced by `__parallel_partial_sort()`.

**4.6.4.28** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_partial_sort ( _RAIter __begin, _RAIter  
__middle, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::partial_sort()`.

#### Parameters

- `__begin` Begin iterator of input sequence.
- `__middle` Sort until this position.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

**4.6.4.29** `template<typename _IIter, typename _OutputIterator  
, typename _BinaryOperation> _OutputIterator  
__gnu_parallel::__parallel_partial_sum ( _IIter __begin, _IIter  
__end, _OutputIterator __result, _BinaryOperation __bin_op )`

Parallel partial sum front-`__end`.

**Parameters**

***\_\_begin*** Begin iterator of input sequence.

***\_\_end*** End iterator of input sequence.

***\_\_result*** Begin iterator of output sequence.

***\_\_bin\_op*** Associative binary function.

**Returns**

End iterator of output sequence.

Definition at line 203 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.30** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::_parallel_partial_sum_basecase ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::value_type __value )`

Base case prefix sum routine.

**Parameters**

***\_\_begin*** Begin iterator of input sequence.

***\_\_end*** End iterator of input sequence.

***\_\_result*** Begin iterator of output sequence.

***\_\_bin\_op*** Associative binary function.

***\_\_value*** Start value. Must be passed since the neutral element is unknown in general.

**Returns**

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.



**4.6.4.31** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum_linear ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::difference_type __n )`

Parallel partial sum implementation, two-phase approach, no recursion.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__result` Begin iterator of output sequence.  
`__bin_op` Associative binary function.  
`__n` Length of sequence.  
`__num_threads` Number of threads to use.

#### Returns

End iterator of output sequence.

Definition at line 90 of file `partial_sum.h`.

References `__parallel_partial_sum_basecase()`, `equally_split()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

**4.6.4.32** `template<typename _RAIter , typename _Predicate  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__parallel_partition ( _RAIter __begin, _RAIter  
__end, _Predicate __pred, _ThreadIndex __num_threads )`

Parallel implementation of `std::partition`.

#### Parameters

`__begin` Begin iterator of input sequence to split.  
`__end` End iterator of input sequence to split.  
`__pred` Partition predicate, possibly including some kind of pivot.  
`__num_threads` Maximum number of threads to use for this task.

#### Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file partition.h.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::partition_chunk_share`, `__gnu_parallel::_Settings::partition_chunk_size`, and `std::swap()`.

Referenced by `__parallel_nth_element()`, `__parallel_sort_qs_divide()`, and `__qsb_divide()`.

```
4.6.4.33 template<typename _RAIter , typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle (_RAIter
__begin, _RAIter __end, _RandomNumberGenerator __rng =
__RandomNumber ()) [inline]
```

Parallel random public call.

#### Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__rng` Random number generator to use.

Definition at line 517 of file random\_shuffle.h.

References `__parallel_random_shuffle_drs()`.

```
4.6.4.34 template<typename _RAIter , typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter
__begin, _RAIter __end, typename std::iterator_traits<
_RAIter >::difference_type __n, _ThreadIndex __num_threads,
_RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

#### Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__n` Length of sequence.
- `__num_threads` Number of threads to use.
- `__rng` Random number generator to use.

Definition at line 263 of file random\_shuffle.h.

References \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_bins\_end, \_\_parallel\_random\_shuffle\_drs\_pu(), \_\_rd\_log2(), \_\_round\_up\_to\_pow2(), \_\_sequential\_random\_shuffle(), \_GLIBCXX\_CALL, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_bin\_proc, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_bins\_begin, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_dist, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bins, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bits, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_num\_threads, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_sd, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_seed, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_starts, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_temporaries, \_\_gnu\_parallel::\_Settings::L2\_cache\_size, std::min(), and \_\_gnu\_parallel::\_Settings::TLB\_size.

Referenced by \_\_parallel\_random\_shuffle().

**4.6.4.35** `template<typename _RAIter, typename _RandomNumberGenerator  
> void __gnu_parallel::_parallel_random_shuffle_drs_pu (  
_DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus )`

Random shuffle code executed by each thread.

#### Parameters

*\_\_pus* Array of thread-local data records.

Definition at line 122 of file random\_shuffle.h.

References \_\_random\_number\_pow2(), \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_dist, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bins, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bits, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_num\_threads, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_sd, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_seed, and \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_starts.

Referenced by \_\_parallel\_random\_shuffle\_drs().

**4.6.4.36** `template<bool __stable, typename _RAIter, typename _Compare >  
void __gnu_parallel::_parallel_sort ( _RAIter __begin, _RAIter  
__end, _Compare __comp, parallel_tag __parallelism ) [inline]`

Choose a parallel sorting algorithm.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.  
***stable*** Sort `stable`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `GLIBCXX_CALL`, and `__gnu_parallel::Settings::get()`.

[illegible]

Choose balanced quicksort for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.  
***\_\_stable*** Sort *\_\_stable*.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `GLIBCXX_CALL`.

[illegible]

Choose multiway mergesort with splitting by sampling, for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

```
graph LR; A["__gnu_parallel::__parallel_sort"] --> B["__gnu_parallel::parallel_tag::__get_num_threads"]
```

**4.6.4.39** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, multiway_mergesort_tag __parallelism )  
 [inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

#### Parameters

**\_\_begin** Begin iterator of input sequence.

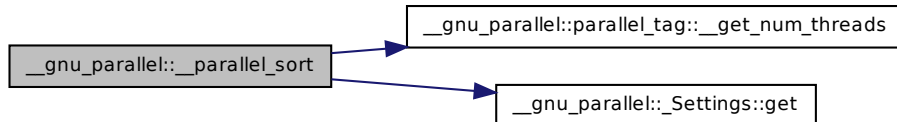
**\_\_end** End iterator of input sequence.

**\_\_comp** Comparator.

Definition at line 74 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`,  
 and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



**4.6.4.40** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, multiway_mergesort_exact_tag  
 __parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

**\_\_begin** Begin iterator of input sequence.

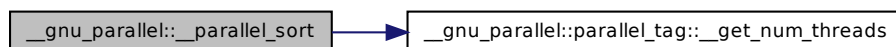
**\_\_end** End iterator of input sequence.

**\_\_comp** Comparator.

Definition at line 97 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



```

4.6.4.41 template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter
__end, _Compare __comp, quicksort_tag __parallelism)
[inline]

```

Choose quicksort for parallel sorting.

#### Parameters

`__begin` Begin iterator of input sequence.

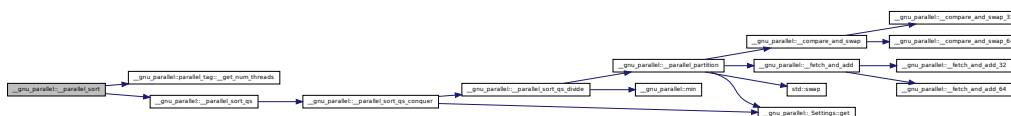
`__end` End iterator of input sequence.

`__comp` Comparator.

Definition at line 136 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.42** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, default_parallel_tag __parallelism )  
 [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

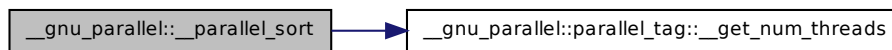
#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__comp` Comparator.

Definition at line 178 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.43** `template<typename _RAIter, typename _Compare > void  
 __gnu_parallel::__parallel_sort_qs ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort main call.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator input sequence, ignored.  
`__comp` Comparator.  
`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 152 of file `quicksort.h`.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.



**4.6.4.44** `template<typename _RAIter, typename _Compare> void  
 __gnu_parallel::__parallel_sort_qs_conquer ( _RAIter __begin,  
 _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort conquer step.

#### Parameters

`__begin` Begin iterator of subsequence.  
`__end` End iterator of subsequence.  
`__comp` Comparator.  
`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 99 of file quicksort.h.

References `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`.

**4.6.4.45** `template<typename _RAIter, typename _Compare  
 > std::iterator_traits<_RAIter>::difference_type  
 __gnu_parallel::__parallel_sort_qs_divide ( _RAIter __begin,  
 _RAIter __end, _Compare __comp, typename std::iterator_traits<  
 _RAIter>::difference_type __pivot_rank, typename  
 std::iterator_traits<_RAIter>::difference_type __num_samples,  
 _ThreadIndex __num_threads )`

Unbalanced quicksort divide step.

#### Parameters

`__begin` Begin iterator of subsequence.  
`__end` End iterator of subsequence.  
`__comp` Comparator.  
`__pivot_rank` Desired \_\_rank of the pivot.  
`__num_samples` Choose pivot from that many samples.  
`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References `__parallel_partition()`, and `min()`.

Referenced by `__parallel_sort_qs_conquer()`.

**4.6.4.46** `template<typename _RAIter, typename _Compare> void  
     __gnu_parallel::__parallel_sort_qsb ( _RAIter __begin, _RAIter  
     __end, _Compare __comp, _ThreadIndex __num_threads )`

Top-level quicksort routine.

#### Parameters

***\_\_begin*** Begin iterator of sequence.  
     ***\_\_end*** End iterator of sequence.  
     ***\_\_comp*** Comparator.  
     ***\_\_num\_threads*** Number of threads that are allowed to work on this part.

Definition at line 430 of file `balanced_quicksort.h`.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

**4.6.4.47** `template<typename _IIter, class _OutputIterator> _OutputIterator  
     __gnu_parallel::__parallel_unique_copy ( _IIter __first, _IIter  
     __last, _OutputIterator __result ) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.

#### Parameters

***\_\_first*** Begin iterator of input sequence.  
     ***\_\_last*** End iterator of input sequence.  
     ***\_\_result*** Begin iterator of result \_\_sequence.

#### Returns

End iterator of result \_\_sequence.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

**4.6.4.48** `template<typename _IIter, class _OutputIterator, class  
     _BinaryPredicate> _OutputIterator __gnu_parallel::__parallel_  
     unique_copy ( _IIter __first, _IIter __last, _OutputIterator __result,  
     _BinaryPredicate __binary_pred )`

Parallel `std::unique_copy()`, w/\_o explicit equality predicate.

**Parameters**

*\_\_first* Begin iterator of input sequence.

*\_\_last* End iterator of input sequence.

*\_\_result* Begin iterator of result *\_\_sequence*.

*\_\_binary\_pred* Equality predicate.

**Returns**

End iterator of result *\_\_sequence*.

Definition at line 50 of file `unique_copy.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

Referenced by `__parallel_unique_copy()`.

**4.6.4.49** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__qsb_conquer ( _QSBThreadLocal< _RAIter >  
** __tls, _RAIter __begin, _RAIter __end, _Compare __comp,  
_ThreadIndex __iam, _ThreadIndex __num_threads, bool  
__parent_wait )`

Quicksort conquer step.

**Parameters**

*\_\_tls* Array of thread-local storages.

*\_\_begin* Begin iterator of subsequence.

*\_\_end* End iterator of subsequence.

*\_\_comp* Comparator.

*\_\_iam* Number of the thread processing this function.

*\_\_num\_threads* Number of threads that are allowed to work on this part.

Definition at line 171 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover`, and `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial`.

Referenced by `__parallel_sort_qsb()`.

**4.6.4.50** `template<typename _RAIter, typename _Compare  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end,  
_Compare __comp, _ThreadIndex __num_threads )`

Balanced quicksort divide step.

#### Parameters

`__begin` Begin iterator of subsequence.  
`__end` End iterator of subsequence.  
`__comp` Comparator.  
`__num_threads` Number of threads that are allowed to work on this part.

#### Precondition

`(__end-__begin)>=1`

Definition at line 100 of file `balanced_quicksort.h`.

References `__median_of_three_iterators()`, `__parallel_partition()`, and `std::swap()`.

Referenced by `__qsb_conquer()`.

**4.6.4.51** `template<typename _RAIter, typename _Compare > void  
__gnu_parallel::__qsb_local_sort_with_helping ( _QSBThreadLocal<  
_RAIter > ** __tls, _Compare & __comp, _ThreadIndex __iam,  
bool __wait )`

Quicksort step doing load-balanced local sort.

#### Parameters

`__tls` Array of thread-local storages.  
`__comp` Comparator.  
`__iam` Number of the thread processing this function.

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::__M_elements_leftover`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::__M_initial`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::__M_leftover_parts`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::__M_num_threads`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_front()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front()`, `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`, and `std::swap()`.

Referenced by `__qsb_conquer()`.

**4.6.4.52** `template<typename _RandomNumberGenerator > int  
__gnu_parallel::__random_number_pow2 ( int __logp,  
_RandomNumberGenerator & __rng ) [inline]`

Generate a random number in  $[0, 2^{\text{__logp}})$ .

#### Parameters

*\_\_logp* Logarithm (basis 2) of the upper range *\_\_bound*.

*\_\_rng* Random number generator to use.

Definition at line 115 of file `random_shuffle.h`.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

**4.6.4.53** `template<typename _Size > _Size __gnu_parallel::__rd_log2 ( _Size  
__n ) [inline]`

Calculates the rounded-down logarithm of *\_\_n* for base 2.

#### Parameters

*\_\_n* Argument.

#### Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file `parallel/base.h`.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.54** `template<typename _Tp > _Tp __gnu_parallel::__round_up_to_pow2  
( _Tp __x )`

Round up to the next greater power of 2.

#### Parameters

*\_\_x* Integer to round up

Definition at line 246 of file `random_shuffle.h`.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

**4.6.4.55** `template<typename __RAIter1 , typename __RAIter2 , typename  
__Pred > __RAIter1 __gnu_parallel::__search_template ( __RAIter1  
__begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2  
__end2, __Pred __pred )`

Parallel `std::search`.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__pred` Find predicate.

#### Returns

Place of finding in first sequences.

Definition at line 81 of file `search.h`.

References `__calc_borders()`, `_GLIBCXX_CALL`, `equally_split()`, and `min()`.

**4.6.4.56** `template<bool __stable, bool __sentinels, typename _RAIterIterator ,  
typename _RAIter3 , typename _DifferenceTp , typename _Compare  
> _RAIter3 __gnu_parallel::__sequential_multiway_merge (   
_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< _RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, _DifferenceTp __length, _Compare  
__comp )`

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and run-time settings.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.  
`__seqs_end` End iterator of iterator pair input sequence.  
`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, possibly larger than the number of elements available.

`__stable` Stable merging incurs a performance penalty.

`__sentinel` The sequences have `__a` `__sentinel` element.

### Returns

End iterator of output sequence.

Definition at line 917 of file `multiway_merge.h`.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `__GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

**4.6.4.57** `template<typename _RAIter, typename _RandomNumberGenerator>  
> void __gnu_parallel::__sequential_random_shuffle ( _RAIter  
__begin, _RAIter __end, _RandomNumberGenerator & __rng )`

Sequential cache-efficient random shuffle.

### Parameters

`__begin` Begin iterator of sequence.

`__end` End iterator of sequence.

`__rng` Random number generator to use.

Definition at line 408 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.58** `template<typename _Iter> void __gnu_parallel::__shrink (   
std::vector< _Iter> & __os_starts, size_t & __count_to_two, size_t  
& __range_length )`

Combines two ranges into one and thus halves the number of ranges.

### Parameters

`__os_starts` Start positions worked on (oversampled).

***\_\_count\_to\_two*** Counts up to 2.  
***\_\_range\_length*** Current length of a chunk.

Definition at line 70 of file list\_partition.h.

References `std::vector< _Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

**4.6.4.59** `template<typename _Iter > void __gnu_parallel::__shrink_and_double ( std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice )`

Shrinks and doubles the ranges.

#### Parameters

***\_\_os\_starts*** Start positions worked on (oversampled).  
***\_\_count\_to\_two*** Counts up to 2.  
***\_\_range\_length*** Current length of a chunk.  
***\_\_make\_twice*** Whether the `__os_starts` is allowed to be grown or not

Definition at line 50 of file list\_partition.h.

References `__shrink()`, `std::vector< _Tp, _Alloc >::resize()`, and `std::vector< _Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

**4.6.4.60** `void __gnu_parallel::__yield ( ) [inline]`

Yield the control to another thread, without waiting for the end to the time slice.

Definition at line 354 of file parallel/compatibility.h.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

**4.6.4.61** `template<typename _DifferenceType, typename _OutputIterator > _OutputIterator __gnu_parallel::equally_split ( _DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s )`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.



**Parameters**

`__n` Number of elements  
`__num_threads` Number of parts  
`__s` Splitters

**Returns**

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

**4.6.4.62** `template<typename _DifferenceType > _DifferenceType  
 __gnu_parallel::equally_split_point ( _DifferenceType __n,  
 _ThreadIndex __num_threads, _ThreadIndex __thread_no )`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

**Parameters**

`__n` Number of elements  
`__num_threads` Number of parts

**Returns**

splitting point

Definition at line 74 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

**4.6.4.63** `template<typename _IIter , typename _FunctorType > size_t  
 __gnu_parallel::list_partition ( const _IIter __begin, const _IIter  
 __end, _IIter * __starts, size_t * __lengths, const int __num_parts,  
 _FunctorType & __f, int __oversampling = 0 )`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

**Parameters**

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__starts` Start iterators for the resulting parts, dimension `__num_parts+1`. For convenience, `__starts [__num_parts]` contains the end iterator of the sequence.
- `__lengths` Length of the resulting parts.
- `__num_parts` Number of parts to split the sequence into.
- `__f` Functor to be applied to each element by traversing `__it`
- `__oversampling` Oversampling factor. If 0, then the partitions will differ in at most  $\{ \{__end\} - \{__begin\} \} \_elements$ . Otherwise, the ratio between the longest and the shortest part is bounded by  $1/(\{__oversampling\} \{num\})$

**Returns**

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector<_Tp, _Alloc>::size()`.

**4.6.4.64** `template<typename _Tp> const _Tp& __gnu_parallel::max ( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

Referenced by `__parallel_nth_element()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.65** `template<typename _Tp> const _Tp& __gnu_parallel::min ( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`, `__parallel_sort_qs_divide()`, `__search_template()`, `multiseq_partition()`, and `multiseq_selection()`.

```

4.6.4.66 template<typename _RanSeqs , typename _RankType ,
 typename _RankIterator , typename _Compare > void __gnu_
 parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs
 __end_seqs, _RankType __rank, _RankIterator __begin_offsets,
 _Compare __comp = std::less< typename std::iterator_
 traits<typename std::iterator_traits<_RanSeqs>::value_
 type:: first_type>::value_type> ()
)

```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

#### Parameters

`__begin_seqs` Begin of the sequence of iterator pairs.

`__end_seqs` End of the sequence of iterator pairs.

`__rank` The global rank to partition at.

`__begin_offsets` A random-access `__sequence` `__begin` where the `__result` will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective `__sequence`.

`__comp` The ordering functor, defaults to `std::less<_Tp>`.

Definition at line 124 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `max()`, `min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

Referenced by `multiway_merge_exact_splitting()`.

```

4.6.4.67 template<typename _Tp , typename _RanSeqs , typename _RankType
 , typename _Compare > _Tp __gnu_parallel::multiseq_selection (
 _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,
 _RankType & __offset, _Compare __comp = std::less<_Tp> ())

```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

**Parameters**

- `__begin_seqs`** Begin of the sequence of iterator pairs.
- `__end_seqs`** End of the sequence of iterator pairs.
- `__rank`** The global rank to partition at.
- `__offset`** The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
- `__comp`** The ordering functor, defaults to [std::less](#).

Definition at line 390 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `max()`, `min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

**4.6.4.68** `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut  
__gnu_parallel::multiway_merge ( _RAIterPairIterator  
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
__target, _DifferenceTp __length, _Compare __comp,  
__gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __i < 10; ++__j)
 sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

#### See also

`stable_multiway_merge`

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

#### Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

#### Parameters

*`_RAIterPairIterator`* iterator over sequence of pairs of iterators

*`_RAIterOut`* iterator over target sequence

*`_DifferenceTp`* difference type for the sequence

*`_Compare`* strict weak ordering type to compare elements in sequences

*`__seqs_begin`* `__begin` of sequence `__sequence`

*`__seqs_end`* `_M_end` of sequence `__sequence`

*`__target`* target sequence to merge to.

*`__comp`* strict weak ordering to use for element comparison.

*`__length`* Maximum length to merge, possibly larger than the number of elements available.

#### Returns

`_M_end` iterator of output sequence

Definition at line 1410 of file multiway\_merge.h.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**4.6.4.69** `template<template< typename RAI, typename C > class  
iterator, typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_3_variant ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 234 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.70** `template<template< typename RAI, typename C > class  
iterator, typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_4_variant ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

- `__seqs_begin` Begin iterator of iterator pair input sequence.
- `__seqs_end` End iterator of iterator pair input sequence.
- `__target` Begin iterator of output sequence.
- `__comp` Comparator.
- `__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

- End iterator of output sequence.

Definition at line 353 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**4.6.4.71** `template<bool __stable, typename _RAIterIterator ,  
typename _Compare , typename _DifferenceType > void  
__gnu_parallel::multiway_merge_exact_splitting ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _DifferenceType  
__length, _DifferenceType __total_length, _Compare __comp,  
std::vector< std::pair< _DifferenceType, _DifferenceType > > *  
__pieces )`

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1112 of file multiway\_merge.h.

References `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equally_split()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.72** `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_loser_tree ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 484 of file multiway\_merge.h.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.



**4.6.4.73** `template<typename UnguardedLoserTree , typename _RAIterIterator  
, typename _RAIter3 , typename _DifferenceTp , typename _Compare  
> _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel  
( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< _RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, _DifferenceTp __length, _Compare  
__comp )`

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

#### Parameters

- \_\_stable* The value must be the same as for the used LoserTrees.
- UnguardedLoserTree* \_Loser Tree variant to use for the unguarded merging.
- GuardedLoserTree* \_Loser Tree variant to use for the guarded merging.
- \_\_seqs\_begin* Begin iterator of iterator pair input sequence.
- \_\_seqs\_end* End iterator of iterator pair input sequence.
- \_\_target* Begin iterator of output sequence.
- \_\_comp* Comparator.
- \_\_length* Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 658 of file multiway\_merge.h.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

**4.6.4.74** `template<typename _LT , typename _RAIterIterator , typename  
_RAIter3 , typename _DifferenceTp , typename _Compare >  
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded  
( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< _RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, _DifferenceTp __length, _Compare  
__comp )`

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the LoserTree class `_LT`.

Stability is selected by the used LoserTrees.

**Precondition**

No input will run out of elements during the merge.

**Parameters**

***\_\_seqs\_begin*** Begin iterator of iterator pair input sequence.  
***\_\_seqs\_end*** End iterator of iterator pair input sequence.  
***\_\_target*** Begin iterator of output sequence.  
***\_\_comp*** Comparator.  
***\_\_length*** Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 567 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.75** `template<bool __stable, typename _RAIterIterator ,  
 typename _Compare , typename _DifferenceType > void  
 __gnu_parallel::multiway_merge_sampling_splitting (   
 _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
 _DifferenceType __length, _DifferenceType __total_length,  
 _Compare __comp, std::vector< std::pair< _DifferenceType,  
 _DifferenceType > > * __pieces )`

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1032 of file multiway\_merge.h.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`,  
 and `__gnu_parallel::_Settings::merge_oversampling`.

**4.6.4.76** `template<typename _RAIterPairIterator , typename _RAIterOut ,  
 typename _DifferenceTp , typename _Compare > _RAIterOut  
 __gnu_parallel::multiway_merge_sentinels ( _RAIterPairIterator  
 __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
 __target, _DifferenceTp __length, _Compare __comp,  
 __gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_`-  
`begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an

iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 11; ++__j)
 sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

### Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

**See also**

`stable_multiway_merge_sentinels`

**Parameters**

***\_RAIterPairIterator*** iterator over sequence of pairs of iterators  
***\_RAIterOut*** iterator over target sequence  
***\_DifferenceTp*** difference type for the sequence  
***\_Compare*** strict weak ordering type to compare elements in sequences  
***\_\_seqs\_begin*** \_\_begin of sequence \_\_sequence  
***\_\_seqs\_end*** \_\_M\_end of sequence \_\_sequence  
***\_\_target*** target sequence to merge to.  
***\_\_comp*** strict weak ordering to use for element comparison.  
***\_\_length*** Maximum length to merge, possibly larger than the number of elements available.

**Returns**

`__M_end` iterator of output sequence

Definition at line 1774 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**4.6.4.77** `template<bool __stable, bool __sentinels, typename  
     _RAIterIterator, typename _RAIter3, typename _DifferenceTp  
     , typename _Splitter, typename _Compare > _RAIter3  
     __gnu_parallel::parallel_multiway_merge ( _RAIterIterator  
     __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
     _Splitter __splitter, _DifferenceTp __length, _Compare __comp,  
     _ThreadIndex __num_threads )`

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and run-time settings.

Must not be called if the number of sequences is 1.

**Parameters**

***\_Splitter*** functor to split input (either `__exact` or sampling based)  
***\_\_seqs\_begin*** Begin iterator of iterator pair input sequence.  
***\_\_seqs\_end*** End iterator of iterator pair input sequence.

- \_\_*target* Begin iterator of output sequence.
- \_\_*comp* Comparator.
- \_\_*length* Maximum length to merge, possibly larger than the number of elements available.
- \_\_*stable* Stable merging incurs a performance penalty.
- \_\_*sentinel* Ignored.

### Returns

End iterator of output sequence.

Definition at line 1217 of file multiway\_merge.h.

References \_\_is\_sorted(), \_\_GLIBCXX\_CALL, \_\_GLIBCXX\_PARALLEL\_LENGTH, \_\_gnu\_parallel::\_Settings::get(), std::make\_pair(), and \_\_gnu\_parallel::\_Settings::merge\_oversampling.

Referenced by \_\_parallel\_merge\_advance().

**4.6.4.78** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

PMWMS main call.

### Parameters

- \_\_*begin* Begin iterator of sequence.
- \_\_*end* End iterator of sequence.
- \_\_*comp* Comparator.
- \_\_*n* Length of sequence.
- \_\_*num\_threads* Number of threads to use.

Definition at line 394 of file multiway\_mergesort.h.

References \_\_GLIBCXX\_CALL, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_num\_threads, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_offsets, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_pieces, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_samples, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_source, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_starts, \_\_gnu\_parallel::\_PMWMSortingData< \_RAIter >::\_M\_temporary, \_\_gnu\_parallel::\_Settings::get(), std::vector< \_Tp, \_Alloc >::resize(), and \_\_gnu\_parallel::\_Settings::sort\_mwms\_oversampling.

**4.6.4.79** `template<bool __stable, bool __exact, typename _RAIter, typename  
_Compare > void __gnu_parallel::parallel_sort_mwms_pu (  
_PMWMSortingData< _RAIter > * __sd, _Compare & __comp )`

PMWMS code executed by each thread.

#### Parameters

`__sd` Pointer to algorithm data.

`__comp` Comparator.

Definition at line 308 of file multiway\_mergesort.h.

References `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_num_`  
threads, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces`,  
`__gnu_parallel::_PMWMSortingData< _RAIter >::_M_source`, `__gnu_`  
`parallel::_PMWMSortingData< _RAIter >::_M_starts`, `__gnu_parallel::_`  
`PMWMSortingData< _RAIter >::_M_temporary`, `__gnu_parallel::_Settings::get()`,  
`std::make_pair()`, `__gnu_parallel::_Settings::sort_mwms_oversampling`, and  
`std::uninitialized_copy()`.

### 4.6.5 Variable Documentation

**4.6.5.1** `const int __gnu_parallel::_CASable_bits [static]`

Number of bits of `_CASable`.

Definition at line 130 of file types.h.

Referenced by `__decode2()`, and `__encode2()`.

**4.6.5.2** `const _CASable __gnu_parallel::_CASable_mask [static]`

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file types.h.

Referenced by `__decode2()`.

## 4.7 \_\_gnu\_pbds Namespace Reference

GNU extensions for policy-based data structures for public use.

## Classes

- struct [associative\\_container\\_tag](#)  
*Basic associative-container.*
- class [basic\\_hash\\_table](#)  
*An abstract basic hash-based associative container.*
- struct [basic\\_hash\\_tag](#)  
*Basic hash.*
- class [basic\\_tree](#)  
*An abstract basic tree-like (tree, trie) associative container.*
- struct [basic\\_tree\\_tag](#)  
*Basic tree.*
- struct [binary\\_heap\\_tag](#)  
*Binary-heap (array-based).*
- struct [binomial\\_heap\\_tag](#)  
*Binomial-heap.*
- class [cc\\_hash\\_table](#)  
*A concrete collision-chaining hash-based associative container.*
- struct [cc\\_hash\\_tag](#)  
*Collision-chaining hash.*
- class [container\\_base](#)  
*An abstract basic associative container.*
- struct [container\\_tag](#)  
*Base data structure tag.*
- struct [container\\_traits](#)  
*[container\\_traits](#)*
- class [gp\\_hash\\_table](#)  
*A concrete general-probing hash-based associative container.*
- struct [gp\\_hash\\_tag](#)  
*General-probing hash.*

- class [list\\_update](#)  
*A list-update based associative container.*
- struct [list\\_update\\_tag](#)  
*List-update.*
- struct [null\\_mapped\\_type](#)  
*A mapped-policy indicating that an associative container is a set.*
- struct [ov\\_tree\\_tag](#)  
*Ordered-vector tree.*
- struct [pairing\\_heap\\_tag](#)  
*Pairing-heap.*
- struct [pat\\_trie\\_tag](#)  
*PATRICIA trie.*
- struct [priority\\_queue\\_tag](#)  
*Basic priority-queue.*
- struct [rb\\_tree\\_tag](#)  
*Red-black tree.*
- struct [rc\\_binomial\\_heap\\_tag](#)  
*Redundant-counter binomial-heap.*
- struct [sequence\\_tag](#)  
*Basic sequence.*
- struct [splay\\_tree\\_tag](#)  
*Splay tree.*
- struct [string\\_tag](#)  
*Basic string container, inclusive of strings, ropes, etc.*
- struct [thin\\_heap\\_tag](#)  
*Thin heap.*
- class [tree](#)  
*A concrete basic tree-based associative container.*



- struct [tree\\_tag](#)  
*tree.*
- class [trie](#)  
*A concrete basic trie-based associative container.*
- struct [trie\\_tag](#)  
*trie.*

## Typedefs

- typedef void `trivial_iterator_difference_type`

## Functions

- void `__throw_container_error` (void)
- void `__throw_insert_error` (void)
- void `__throw_join_error` (void)
- void `__throw_resize_error` (void)

### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 `__gnu_profile` Namespace Reference

GNU profile code for public use.

## Classes

- class [\\_\\_container\\_size\\_info](#)  
*A container size instrumentation line in the object table.*
- class [\\_\\_container\\_size\\_stack\\_info](#)  
*A container size instrumentation line in the stack table.*
- class [\\_\\_hashfunc\\_info](#)

*A hash performance instrumentation line in the object table.*

- class [\\_\\_hashfunc\\_stack\\_info](#)

*A hash performance instrumentation line in the stack table.*

- class [\\_\\_list2vector\\_info](#)

*A list-to-vector instrumentation line in the object table.*

- class [\\_\\_map2umap\\_info](#)

*A map-to-unordered\_map instrumentation line in the object table.*

- class [\\_\\_map2umap\\_stack\\_info](#)

*A map-to-unordered\_map instrumentation line in the stack table.*

- class [\\_\\_object\\_info\\_base](#)

*Base class for a line in the object table.*

- struct [\\_\\_reentrance\\_guard](#)

*Reentrance guard.*

- class [\\_\\_stack\\_hash](#)

*Hash function for summary trace using call stack as index.*

- class [\\_\\_stack\\_info\\_base](#)

*Base class for a line in the stack table.*

- class [\\_\\_trace\\_base](#)

*Base class for all trace producers.*

- class [\\_\\_trace\\_container\\_size](#)

*Container size instrumentation trace producer.*

- class [\\_\\_trace\\_hash\\_func](#)

*Hash performance instrumentation producer.*

- class [\\_\\_trace\\_hashtable\\_size](#)

*Hashtable size instrumentation trace producer.*

- class [\\_\\_trace\\_map2umap](#)

*Map-to-unordered\_map instrumentation producer.*

- class [\\_\\_trace\\_vector\\_size](#)

*Hashtable size instrumentation trace producer.*

- class `__trace_vector_to_list`  
*Vector-to-list instrumentation producer.*
- class `__vector2list_info`  
*A vector-to-list instrumentation line in the object table.*
- class `__vector2list_stack_info`  
*A vector-to-list instrumentation line in the stack table.*
- struct `__warning_data`  
*Representation of a warning.*

## Typedefs

- typedef `std::vector< __cost_factor * >` `__cost_factor_vector`
- typedef `std::unordered_map< std::string, std::string >` `__env_t`
- typedef `void *` `__instruction_address_t`
- typedef `const void *` `__object_t`
- typedef `std::vector< __instruction_address_t >` `__stack_npt`
- typedef `__stack_npt *` `__stack_t`
- typedef `std::vector< __warning_data >` `__warning_vector_t`

## Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

## Functions

- `std::size_t __env_to_size_t (const char * __env_var, std::size_t __default_value)`
- `template<typename _InputIterator, typename _Function >`  
`_Function __for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `__cost_factor_vector *& __get__cost_factors ()`
- `__env_t & __get__env ()`
- `__gnu_cxx::__mutex & __get__global_lock ()`
- `__cost_factor & __get__list_iterate_cost_factor ()`
- `__cost_factor & __get__list_resize_cost_factor ()`
- `__cost_factor & __get__list_shift_cost_factor ()`

- `__cost_factor & __get__map_erase_cost_factor ()`
- `__cost_factor & __get__map_find_cost_factor ()`
- `__cost_factor & __get__map_insert_cost_factor ()`
- `__cost_factor & __get__map_iterate_cost_factor ()`
- `__cost_factor & __get__umap_erase_cost_factor ()`
- `__cost_factor & __get__umap_find_cost_factor ()`
- `__cost_factor & __get__umap_insert_cost_factor ()`
- `__cost_factor & __get__umap_iterate_cost_factor ()`
- `__cost_factor & __get__vector_iterate_cost_factor ()`
- `__cost_factor & __get__vector_resize_cost_factor ()`
- `__cost_factor & __get__vector_shift_cost_factor ()`
- `__trace_hash_func *& __get__S_hash_func ()`
- `__trace_hashtable_size *& __get__S_hashtable_size ()`
- `__trace_list_to_slist *& __get__S_list_to_slist ()`
- `__trace_list_to_vector *& __get__S_list_to_vector ()`
- `__trace_map2umap *& __get__S_map2umap ()`
- `std::size_t & __get__S_max_mem ()`
- `std::size_t & __get__S_max_stack_depth ()`
- `std::size_t & __get__S_max_warn_count ()`
- `const char *& __get__S_trace_file_name ()`
- `__trace_vector_size *& __get__S_vector_size ()`
- `__trace_vector_to_list *& __get__S_vector_to_list ()`
- `__stack_t __get_stack ()`
- `template<typename _Container >  
void __insert_top_n (_Container &__output, const typename _-  
Container::value_type &__value, typename _Container::size_type __n)`
- `bool __is_invalid ()`
- `bool __is_off ()`
- `bool __is_on ()`
- `int __log2 (std::size_t __size)`
- `int __log_magnitude (float __f)`
- `float __map_erase_cost (std::size_t __size)`
- `float __map_find_cost (std::size_t __size)`
- `float __map_insert_cost (std::size_t __size)`
- `std::size_t __max_mem ()`
- `FILE * __open_output_file (const char *__extension)`
- `bool __profcxx_init ()`
- `void __profcxx_init_unconditional ()`
- `void __read_cost_factors ()`
- `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator __remove (_ForwardIterator __first, _ForwardIterator __last,  
const _Tp &__value)`

- void `__report` (void)
- void `__set_cost_factors` ()
- void `__set_max_mem` ()
- void `__set_max_stack_trace_depth` ()
- void `__set_max_warn_count` ()
- void `__set_trace_path` ()
- `std::size_t` `__size` (`__stack_t` `__stack`)
- `std::size_t` `__stack_max_depth` ()
- `template<typename _Container >`  
   void `__top_n` (const `_Container` &`__input`, `_Container` &`__output`, `typename _Container::size_type` `__n`)
- void `__trace_hash_func_construct` (const void \*)
- void `__trace_hash_func_destruct` (const void \*, `std::size_t`, `std::size_t`, `std::size_t`)
- void `__trace_hash_func_init` ()
- void `__trace_hash_func_report` (FILE \*`__f`, `__warning_vector_t` &`__warnings`)
- void `__trace_hashtable_size_construct` (const void \*, `std::size_t`)
- void `__trace_hashtable_size_destruct` (const void \*, `std::size_t`, `std::size_t`)
- void `__trace_hashtable_size_init` ()
- void `__trace_hashtable_size_report` (FILE \*`__f`, `__warning_vector_t` &`__warnings`)
- void `__trace_hashtable_size_resize` (const void \*, `std::size_t`, `std::size_t`)
- void `__trace_list_to_set_construct` (const void \*)
- void `__trace_list_to_set_destruct` (const void \*)
- void `__trace_list_to_set_find` (const void \*, `std::size_t`)
- void `__trace_list_to_set_insert` (const void \*, `std::size_t`, `std::size_t`)
- void `__trace_list_to_set_invalid_operator` (const void \*)
- void `__trace_list_to_set_iterate` (const void \*, `std::size_t`)
- void `__trace_list_to_slist_construct` (const void \*)
- void `__trace_list_to_slist_destruct` (const void \*)
- void `__trace_list_to_slist_init` ()
- void `__trace_list_to_slist_operation` (const void \*)
- void `__trace_list_to_slist_report` (FILE \*`__f`, `__warning_vector_t` &`__warnings`)
- void `__trace_list_to_slist_rewind` (const void \*)
- void `__trace_list_to_vector_construct` (const void \*)
- void `__trace_list_to_vector_destruct` (const void \*)
- void `__trace_list_to_vector_init` ()
- void `__trace_list_to_vector_insert` (const void \*, `std::size_t`, `std::size_t`)
- void `__trace_list_to_vector_invalid_operator` (const void \*)
- void `__trace_list_to_vector_iterate` (const void \*, `std::size_t`)

- void **\_\_trace\_list\_to\_vector\_report** (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void **\_\_trace\_list\_to\_vector\_resize** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_construct** (const void \*)
- void **\_\_trace\_map\_to\_unordered\_map\_destruct** (const void \*)
- void **\_\_trace\_map\_to\_unordered\_map\_erase** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_find** (const void \*, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_init** ()
- void **\_\_trace\_map\_to\_unordered\_map\_insert** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_invalidate** (const void \*)
- void **\_\_trace\_map\_to\_unordered\_map\_iterate** (const void \*, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_report** (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void **\_\_trace\_vector\_size\_construct** (const void \*, std::size\_t)
- void **\_\_trace\_vector\_size\_destruct** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_vector\_size\_init** ()
- void **\_\_trace\_vector\_size\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_trace\_vector\_size\_resize** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_construct** (const void \*)
- void **\_\_trace\_vector\_to\_list\_destruct** (const void \*)
- void **\_\_trace\_vector\_to\_list\_find** (const void \*, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_init** ()
- void **\_\_trace\_vector\_to\_list\_insert** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_invalid\_operator** (const void \*)
- void **\_\_trace\_vector\_to\_list\_iterate** (const void \*, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_trace\_vector\_to\_list\_resize** (const void \*, std::size\_t, std::size\_t)
- bool **\_\_turn** (\_\_state\_type \_\_s)
- bool **\_\_turn\_off** ()
- bool **\_\_turn\_on** ()
- void **\_\_write** (FILE \*\_\_f, \_\_stack\_t \_\_stack)
- void **\_\_write\_cost\_factors** ()
- **\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_state\_type, \_\_state, \_\_INVALID)

### 4.8.1 Detailed Description

GNU profile code for public use.

## 4.8.2 Typedef Documentation

### 4.8.2.1 `typedef std:: std ::unordered_map<std::string, std::string> __gnu_profile::__env_t`

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is libstdcxx-profile.conf. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 72 of file `profiler_trace.h`.

## 4.8.3 Function Documentation

### 4.8.3.1 `__gnu_cxx::__mutex& __gnu_profile::__get__global_lock ( ) [inline]`

Master lock.

Definition at line 77 of file `profiler_trace.h`.

### 4.8.3.2 `bool __gnu_profile::__profcxx_init ( ) [inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 656 of file `profiler_trace.h`.

### 4.8.3.3 `void __gnu_profile::__report ( void ) [inline]`

Final report method, registered with `atexit`.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in `profiler.h`. However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 447 of file `profiler_trace.h`.

References `std::min()`.

## 4.9 `__gnu_sequential` Namespace Reference

GNU sequential classes for public use.

### 4.9.1 Detailed Description

GNU sequential classes for public use.

## 4.10 `abi` Namespace Reference

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

### 4.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`. A brief overview of an ABI is given in the `libstdc++` FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 4.11 `std` Namespace Reference

ISO C++ entities toplevel namespace is `std`.

### Namespaces

- namespace `__debug`
- namespace `__detail`



- namespace [\\_\\_parallel](#)
- namespace [\\_\\_profile](#)
- namespace [chrono](#)
- namespace [decimal](#)
- namespace [placeholders](#)
- namespace [regex\\_constants](#)
- namespace [rel\\_ops](#)
- namespace [this\\_thread](#)
- namespace [tr1](#)

## Classes

- class [\\_\\_basic\\_future](#)  
*Common implementation for future and [shared\\_future](#).*
- class [\\_\\_codecvt\\_abstract\\_base](#)  
*Common base for codecvt functions.*
- class [\\_\\_ctype\\_abstract\\_base](#)  
*Common base for ctype facet.*
- struct [\\_\\_declval\\_protector](#)  
*declval*
- struct [\\_\\_future\\_base](#)  
*Base class and enclosing scope.*
- struct [\\_\\_is\\_location\\_invariant](#)
- struct [\\_\\_numeric\\_limits\\_base](#)  
*Part of [std::numeric\\_limits](#).*
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- struct [\\_Build\\_index\\_tuple](#)  
*Builds an [\\_Index\\_tuple](#)<0, 1, 2, ..., [\\_Num-1](#)>.*
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)  
*A [deque::iterator](#).*
- struct [\\_Derives\\_from\\_binary\\_function](#)

*Determines if the type `_Tp` derives from [binary\\_function](#).*

- [struct `\_Derives\_from\_unary\_function`](#)

*Determines if the type `_Tp` derives from [unary\\_function](#).*

- [class `\_Function\_base`](#)

*Base class of all polymorphic function object wrappers.*

- [struct `\_Function\_to\_function\_pointer`](#)

*Turns a function type into a function pointer type.*

- [struct `\_Fwd\_list\_base`](#)

*Base class for `forward_list`.*

- [struct `\_Fwd\_list\_const\_iterator`](#)

*A `forward_list::const_iterator`.*

- [struct `\_Fwd\_list\_iterator`](#)

*A `forward_list::iterator`.*

- [struct `\_Fwd\_list\_node`](#)

*A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.*

- [struct `\_Fwd\_list\_node\_base`](#)

*A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*

- [class `\_Has\_result\_type\_helper`](#)

- [struct `\_Index\_tuple`](#)

- [class `\_List\_base`](#)

*See [bits/stl\\_deque.h](#)'s [\\_Deque\\_base](#) for an explanation.*

- [struct `\_List\_const\_iterator`](#)

*A `list::const_iterator`.*

- [struct `\_List\_iterator`](#)

*A `list::iterator`.*

- [struct `\_List\_node`](#)

*An actual node in the list.*

- [struct `\_List\_node\_base`](#)

*Common part of a node in the list.*

- struct `_Maybe_get_result_type`  
*If we have found a `result_type`, extract it.*
- struct `_Maybe_unary_or_binary_function`
- struct `_Maybe_unary_or_binary_function< _Res, _T1 >`  
*Derives from `unary_function`, as appropriate.*
- struct `_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`  
*Derives from `binary_function`, as appropriate.*
- struct `_Maybe_wrap_member_pointer`
- struct `_Maybe_wrap_member_pointer< _Tp _Class::* >`
- class `_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >`  
*Implementation of `mem_fn` for const member function pointers.*
- class `_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >`  
*Implementation of `mem_fn` for const volatile member function pointers.*
- class `_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >`  
*Implementation of `mem_fn` for volatile member function pointers.*
- class `_Mem_fn< _Res(_Class::*)(_ArgTypes...) >`  
*Implementation of `mem_fn` for member function pointers.*
- class `_Mu< _Arg, false, false >`
- class `_Mu< _Arg, false, true >`
- class `_Mu< _Arg, true, false >`
- class `_Mu< reference_wrapper< _Tp >, false, false >`
- struct `_Placeholder`  
*The type of placeholder objects defined by `libstdc++`.*
- struct `_Reference_wrapper_base`
- struct `_Safe_tuple_element`
- struct `_Safe_tuple_element_impl`
- struct `_Safe_tuple_element_impl< __i, _Tuple, false >`
- class `_Temporary_buffer`
- struct `_Tuple_impl< _Idx >`
- struct `_Tuple_impl< _Idx, _Head, _Tail... >`
- struct `_Vector_base`  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*

- struct [\\_Weak\\_result\\_type](#)
- struct [\\_Weak\\_result\\_type\\_impl](#)
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(&\)\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function reference.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function type.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Retrieve result type for a const member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Retrieve result type for a const volatile member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Retrieve result type for a volatile member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve result type for a member function pointer.*
- struct [add\\_lvalue\\_reference](#)  
*add\_lvalue\_reference*
- struct [add\\_rvalue\\_reference](#)  
*add\_rvalue\_reference*
- struct [adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct [aligned\\_storage](#)  
*Alignment type.*
- class [allocator](#)  
*The standard allocator, as per [20.4].  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*
- class [allocator< void >](#)  
*allocator<void> specialization.*

- struct `atomic`  
*atomic /// 29.4.3, Generic atomic type, primary class template.*
- struct `atomic< _Tp * >`  
*Partial specialization for pointer types.*
- struct `atomic< bool >`  
*Explicit specialization for bool.*
- struct `atomic< char >`  
*Explicit specialization for char.*
- struct `atomic< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `atomic< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `atomic< int >`  
*Explicit specialization for int.*
- struct `atomic< long >`  
*Explicit specialization for long.*
- struct `atomic< long long >`  
*Explicit specialization for long long.*
- struct `atomic< short >`  
*Explicit specialization for short.*
- struct `atomic< signed char >`  
*Explicit specialization for signed char.*
- struct `atomic< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `atomic< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `atomic< unsigned long >`  
*Explicit specialization for unsigned long.*

- struct [atomic< unsigned long long >](#)  
*Explicit specialization for unsigned long long.*
- struct [atomic< unsigned short >](#)  
*Explicit specialization for unsigned short.*
- struct [atomic< void \\* >](#)  
*Explicit specialization for void\*.*
- struct [atomic< wchar\\_t >](#)  
*Explicit specialization for wchar\_t.*
- class [auto\\_ptr](#)  
*A simple smart pointer providing strict ownership semantics.*
- struct [auto\\_ptr\\_ref](#)
- class [back\\_insert\\_iterator](#)  
*Turns assignment into insertion.*
- class [bad\\_alloc](#)  
*Exception possibly thrown by new.*  
*[bad\\_alloc](#) (or classes derived from it) is used to report allocation errors from the throwing forms of new.*
- class [bad\\_cast](#)  
*Thrown during incorrect typecasting.*  
*If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class [bad\\_exception](#)
- class [bad\\_function\\_call](#)  
*Exception class thrown when class template function's `operator()` is called with an empty target.*
- class [bad\\_typeid](#)  
*Thrown when a NULL pointer in a `typeid` expression is used.*
- class [basic\\_filebuf](#)  
*The actual work of input and output (for files).*  
*This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.*
- class [basic\\_fstream](#)

*Controlling input and output for files.*

*This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class `basic_ifstream`

*Controlling input for files.*

*This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class `basic_ios`

*Virtual base class for all stream classes.*

*Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*

- class `basic_iostream`

*Merging istream and ostream capabilities.*

*This class multiply inherits from the input and output stream classes simply to provide a single interface.*

- class `basic_istream`

*Controlling input.*

*This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.*

- class `basic_istreamstringstream`

*Controlling input for `std::string`.*

*This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*

- class `basic_ofstream`

*Controlling output for files.*

*This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class `basic_ostream`

*Controlling output.*

*This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.*

- class `basic_ostreamstringstream`

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `basic_regex`
- class `basic_streambuf`

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

- class `basic_string`

Managing sequences of characters and character-like objects.

- class `basic_stringbuf`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

- class `basic_stringstream`

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_iostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `bernoulli_distribution`

A Bernoulli random number distribution.

- struct `bidirectional_iterator_tag`

Bidirectional iterators support a superset of forward iterator /// operations.

- struct `binary_function`

- class `binary_negate`

One of the *negation functors*.

- class `binder1st`

One of the *binder functors*.

- class `binder2nd`

One of the *binder functors*.

- class `binomial_distribution`

A discrete binomial random number distribution.



- class [bitset](#)  
*The bitset class represents a fixed-size sequence of bits.*
- class [cauchy\\_distribution](#)  
*A [cauchy\\_distribution](#) random number distribution.*
- struct [char\\_traits](#)  
*Basis for explicit traits specializations.*
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)  
*[char\\_traits<\\_\\_gnu\\_cxx::character>](#) specialization.*
- struct [char\\_traits< char >](#)  
*21.1.3.1 [char\\_traits](#) specializations*
- struct [char\\_traits< wchar\\_t >](#)  
*21.1.3.2 [char\\_traits](#) specializations*
- class [chi\\_squared\\_distribution](#)  
*A [chi\\_squared\\_distribution](#) random number distribution.*
- class [codecvt](#)  
*Primary class template [codecvt](#).  
NB: Generic, mostly useless implementation.*
- class [codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#)  
*[codecvt<InternT, \\_ExternT, encoding\\_state>](#) specialization.*
- class [codecvt< char, char, mbstate\\_t >](#)  
*class [codecvt<char, char, mbstate\\_t>](#) specialization.*
- class [codecvt< wchar\\_t, char, mbstate\\_t >](#)  
*class [codecvt<wchar\\_t, char, mbstate\\_t>](#) specialization.*
- class [codecvt\\_base](#)  
*Empty base class for [codecvt](#) facet [22.2.1.5].*
- class [codecvt\\_byname](#)  
*class [codecvt\\_byname](#) [22.2.1.6].*
- class [collate](#)

*Facet for localized string comparison.*

- class `collate_byname`  
*class `collate_byname` [22.2.4.2].*
- struct `complex`
- class `condition_variable`  
*`condition_variable`*
- class `condition_variable_any`  
*`condition_variable_any`*
- struct `conditional`  
*`conditional`*
- class `const_mem_fun1_ref_t`  
*One of the `adaptors` for member /// pointers.*
- class `const_mem_fun1_t`  
*One of the `adaptors` for member /// pointers.*
- class `const_mem_fun_ref_t`  
*One of the `adaptors` for member /// pointers.*
- class `const_mem_fun_t`  
*One of the `adaptors` for member /// pointers.*
- class `ctype`  
*Primary class template `ctype` facet.  
This template class defines classification and conversion functions for character sets.  
It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.*
- class `ctype< char >`  
*The `ctype<char>` specialization.  
This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.*
- class `ctype< wchar_t >`  
*The `ctype<wchar_t>` specialization.  
This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.*

- struct [ctype\\_base](#)  
*Base class for ctype.*
- class [ctype\\_byname](#)  
*class [ctype\\_byname](#) [22.2.1.2].*
- class [ctype\\_byname< char >](#)  
*22.2.1.4 Class [ctype\\_byname](#) specializations.*
- class [decay](#)  
*decay*
- struct [default\\_delete](#)  
*Primary template, [default\\_delete](#).*
- struct [default\\_delete< \\_Tp\[ \]>](#)  
*Specialization, [default\\_delete](#).*
- struct [defer\\_lock\\_t](#)  
*Do not acquire ownership of the mutex.*
- class [deque](#)  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*
- class [discard\\_block\\_engine](#)
- class [discrete\\_distribution](#)  
*A [discrete\\_distribution](#) random number distribution.*
- struct [divides](#)  
*One of the [math functors](#).*
- class [domain\\_error](#)
- struct [enable\\_if](#)  
*[enable\\_if](#)*
- class [enable\\_shared\\_from\\_this](#)  
*Base class allowing use of member function [shared\\_from\\_this](#).*
- struct [equal\\_to](#)  
*One of the [comparison functors](#).*
- class [error\\_category](#)

*error\_category*

- struct `error_code`

*error\_code*

- struct `error_condition`

*error\_condition*

- class `exception`

*Base class for all library exceptions.*

- class `exponential_distribution`

*An exponential continuous distribution for random numbers.*

- class `extreme_value_distribution`

*A `extreme_value_distribution` random number distribution.*

- class `fisher_f_distribution`

*A `fisher_f_distribution` random number distribution.*

- struct `forward_iterator_tag`

*Forward iterators support a superset of input iterator operations.*

- class `forward_list`

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

- class `fpos`

*Class representing stream positions.*

- class `front_insert_iterator`

*Turns assignment into insertion.*

- class `function<_Res(_ArgTypes...)>`

*Primary class template for `std::function`.  
Polymorphic function wrapper.*

- class `future`

*Primary template for future.*

- class `future<_Res &>`

*Partial specialization for `future<R&>`*

- class `future< void >`  
*Explicit specialization for `future<void>`*
- class `future_error`  
*Exception type thrown by futures.*
- class `gamma_distribution`  
*A gamma continuous distribution for random numbers.*
- class `geometric_distribution`  
*A discrete geometric random number distribution.*
- struct `greater`  
*One of the *comparison functors*.*
- struct `greater_equal`  
*One of the *comparison functors*.*
- class `gslice`  
*Class defining multi-dimensional subset of an array.*
- class `gslice_array`  
*Reference to multi-dimensional subset of an array.*
- struct `has_nothrow_copy_assign`  
*`has_nothrow_copy_assign`*
- struct `has_nothrow_copy_constructor`  
*`has_nothrow_copy_constructor`*
- struct `has_nothrow_default_constructor`  
*`has_nothrow_default_constructor`*
- struct `has_trivial_copy_assign`  
*`has_trivial_copy_assign`*
- struct `has_trivial_copy_constructor`  
*`has_trivial_copy_constructor`*
- struct `has_trivial_default_constructor`  
*`has_trivial_default_constructor`*

- struct [has\\_trivial\\_destructor](#)  
*has\_trivial\_destructor*
- struct [hash](#)  
*Primary class template hash.*
- struct [hash< \\_\\_debug::bitset< \\_Nb > >](#)  
*std::hash specialization for [bitset](#).*
- struct [hash< \\_\\_debug::vector< bool, \\_Alloc > >](#)  
*std::hash specialization for [vector<bool>](#).*
- struct [hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)  
*Explicit specialization of [std::hash](#) for [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#).*
- struct [hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)  
*Explicit specialization of [std::hash](#) for [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#).*
- struct [hash< \\_\\_profile::bitset< \\_Nb > >](#)  
*std::hash specialization for [bitset](#).*
- struct [hash< \\_\\_profile::vector< bool, \\_Alloc > >](#)  
*std::hash specialization for [vector<bool>](#).*
- struct [hash< \\_Tp \\* >](#)  
*Partial specializations for pointer types.*
- struct [hash< error\\_code >](#)  
*std::hash specialization for [error\\_code](#).*
- struct [hash< shared\\_ptr< \\_Tp > >](#)  
*std::hash specialization for [shared\\_ptr](#).*
- struct [hash< string >](#)  
*std::hash specialization for [string](#).*
- struct [hash< thread::id >](#)  
*std::hash specialization for [thread::id](#).*
- struct [hash< u16string >](#)  
*std::hash specialization for [u16string](#).*

- struct `hash< u32string >`  
*std::hash specialization for u32string.*
- struct `hash< unique_ptr< _Tp, _Tp_Deleter > >`  
*std::hash specialization for unique\_ptr.*
- struct `hash< wstring >`  
*std::hash specialization for wstring.*
- struct `hash<::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash<::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- struct `identity`  
*identity*
- class `independent_bits_engine`
- class `indirect_array`  
*Reference to arbitrary subset of an array.*
- class `initializer_list`  
*initializer\_list*
- struct `input_iterator_tag`  
*Marking input iterators.*
- class `insert_iterator`  
*Turns assignment into insertion.*
- class `invalid_argument`
- class `ios_base`  
*The base of the I/O class hierarchy.  
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see ios\_base when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- struct `is_base_of`  
*is\_base\_of*
- struct `is_bind_expression`

*Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].*

- struct `is_bind_expression< _Bind< _Signature > >`  
*Class template `_Bind` is always a bind expression.*
- struct `is_bind_expression< _Bind_result< _Result, _Signature > >`  
*Class template `_Bind` is always a bind expression.*
- struct `is_constructible`  
*`is_constructible`*
- struct `is_convertible`  
*`is_convertible`*
- struct `is_error_code_enum`  
*`is_error_code_enum`*
- struct `is_error_condition_enum`  
*`is_error_condition_enum`*
- struct `is_explicitly_convertible`  
*`is_explicitly_convertible`*
- struct `is_lvalue_reference`  
*`is_lvalue_reference`*
- struct `is_nothrow_constructible`  
*`is_nothrow_constructible`*
- struct `is_placeholder`  
*Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct `is_placeholder< _Placeholder< _Num > >`
- struct `is_pod`  
*`is_pod`*
- struct `is_reference`  
*`is_reference`*
- struct `is_rvalue_reference`  
*`is_rvalue_reference`*



- struct [is\\_signed](#)  
*is\_signed*
- struct [is\\_standard\\_layout](#)  
*is\_standard\_layout*
- struct [is\\_trivial](#)  
*is\_trivial*
- struct [is\\_unsigned](#)  
*is\_unsigned*
- class [istream\\_iterator](#)  
*Provides input iterator semantics for streams.*
- class [istreambuf\\_iterator](#)  
*Provides input iterator semantics for streambufs.*
- struct [iterator](#)  
*Common iterator class.*
- struct [iterator\\_traits](#)  
*Traits class for iterators.*
- struct [iterator\\_traits< \\_Tp \\* >](#)  
*Partial specialization for pointer types.*
- struct [iterator\\_traits< const \\_Tp \\* >](#)  
*Partial specialization for const pointer types.*
- class [length\\_error](#)
- struct [less](#)  
*One of the [comparison functors](#).*
- struct [less\\_equal](#)  
*One of the [comparison functors](#).*
- class [linear\\_congruential\\_engine](#)  
*A model of a linear congruential random number generator.*
- class [list](#)

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

- class [locale](#)

*Container class for localization functionality.*

*The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.*

- class [lock\\_guard](#)

*Scoped lock idiom.*

- class [logic\\_error](#)

*One of two subclasses of exception.*

- struct [logical\\_and](#)

*One of the [Boolean operations functors](#).*

- struct [logical\\_not](#)

*One of the [Boolean operations functors](#).*

- struct [logical\\_or](#)

*One of the [Boolean operations functors](#).*

- class [lognormal\\_distribution](#)

*A [lognormal\\_distribution](#) random number distribution.*

- struct [make\\_signed](#)

*[make\\_signed](#)*

- struct [make\\_unsigned](#)

*[make\\_unsigned](#)*

- class [map](#)

*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

- class [mask\\_array](#)

*Reference to selected subset of an array.*

- class [match\\_results](#)

*The results of a match or search operation.*

- class `mem_fun1_ref_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `mem_fun1_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `mem_fun_ref_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `mem_fun_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `messages`  
*Primary class template [messages](#).  
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct `messages_base`  
*Messages facet base class providing catalog typedef.*
- class `messages_byname`  
*class [messages\\_byname](#) [22.2.7.2].*
- struct `minus`  
*One of the [math functors](#).*
- struct `modulus`  
*One of the [math functors](#).*
- class `money_base`  
*Money format ordering data.  
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the `part` enum. `symbol`, `sign`, and `value` must be present and the remaining field must contain either none or space.*
- class `money_get`  
*Primary class template [money\\_get](#).  
This facet encapsulates the code to parse and return a monetary amount from a string.*
- class `money_put`  
*Primary class template [money\\_put](#).  
This facet encapsulates the code to format and output a monetary amount.*

- class [moneypunct](#)  
*Primary class template [moneypunct](#).  
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*
- class [moneypunct\\_byname](#)  
*class [moneypunct\\_byname](#) [22.2.6.4].*
- class [move\\_iterator](#)
- class [multimap](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- struct [multiplies](#)  
*One of the [math functors](#).*
- class [multiset](#)  
*A standard container made up of elements, which can be retrieved in logarithmic time.*
- class [mutex](#)  
*mutex*
- struct [negate](#)  
*One of the [math functors](#).*
- class [negative\\_binomial\\_distribution](#)  
*A [negative\\_binomial\\_distribution](#) random number distribution.*
- class [nested\\_exception](#)  
*Exception class with [exception\\_ptr](#) data member.*
- class [normal\\_distribution](#)  
*A normal continuous distribution for random numbers.*
- struct [not\\_equal\\_to](#)  
*One of the [comparison functors](#).*
- class [num\\_get](#)  
*Primary class template [num\\_get](#).  
This facet encapsulates the code to parse and return a number from a string. It is used by the [istream](#) numeric extraction operators.*
- class [num\\_put](#)

Primary class template [num\\_put](#).

*This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*

- struct [numeric\\_limits](#)  
*Properties of fundamental types.*
- struct [numeric\\_limits< bool >](#)  
*numeric\_limits<bool> specialization.*
- struct [numeric\\_limits< char >](#)  
*numeric\_limits<char> specialization.*
- struct [numeric\\_limits< char16\\_t >](#)  
*numeric\_limits<char16\_t> specialization.*
- struct [numeric\\_limits< char32\\_t >](#)  
*numeric\_limits<char32\_t> specialization.*
- struct [numeric\\_limits< double >](#)  
*numeric\_limits<double> specialization.*
- struct [numeric\\_limits< float >](#)  
*numeric\_limits<float> specialization.*
- struct [numeric\\_limits< int >](#)  
*numeric\_limits<int> specialization.*
- struct [numeric\\_limits< long >](#)  
*numeric\_limits<long> specialization.*
- struct [numeric\\_limits< long double >](#)  
*numeric\_limits<long double> specialization.*
- struct [numeric\\_limits< long long >](#)  
*numeric\_limits<long long> specialization.*
- struct [numeric\\_limits< short >](#)  
*numeric\_limits<short> specialization.*
- struct [numeric\\_limits< signed char >](#)  
*numeric\_limits<signed char> specialization.*

- struct [numeric\\_limits< unsigned char >](#)  
*numeric\_limits<unsigned char> specialization.*
- struct [numeric\\_limits< unsigned int >](#)  
*numeric\_limits<unsigned int> specialization.*
- struct [numeric\\_limits< unsigned long >](#)  
*numeric\_limits<unsigned long> specialization.*
- struct [numeric\\_limits< unsigned long long >](#)  
*numeric\_limits<unsigned long long> specialization.*
- struct [numeric\\_limits< unsigned short >](#)  
*numeric\_limits<unsigned short> specialization.*
- struct [numeric\\_limits< wchar\\_t >](#)  
*numeric\_limits<wchar\_t> specialization.*
- class [numpunct](#)  
*Primary class template numpunct.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class [numpunct\\_byname](#)  
*class numpunct\_byname [22.2.3.2].*
- struct [once\\_flag](#)  
*once\_flag*
- class [ostream\\_iterator](#)  
*Provides output iterator semantics for streams.*
- class [ostreambuf\\_iterator](#)  
*Provides output iterator semantics for streambufs.*
- class [out\\_of\\_range](#)
- struct [output\\_iterator\\_tag](#)  
*Marking output iterators.*
- class [overflow\\_error](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)

*Partial specialization of `owner_less` for `shared_ptr`.*

- struct `owner_less< weak_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `weak_ptr`.*
- class `packaged_task< _Res(_ArgTypes...)>`  
*`packaged_task`*
- struct `pair`  
*`pair` holds two objects of arbitrary type.*
- class `piecewise_constant_distribution`  
*A `piecewise_constant_distribution` random number distribution.*
- class `piecewise_linear_distribution`  
*A `piecewise_linear_distribution` random number distribution.*
- struct `plus`  
*One of the `math` functors.*
- class `pointer_to_binary_function`  
*One of the `adaptors` for function pointers.*
- class `pointer_to_unary_function`  
*One of the `adaptors` for function pointers.*
- class `poisson_distribution`  
*A discrete Poisson random number distribution.*
- class `priority_queue`  
*A standard container automatically sorting its contents.*
- class `promise`  
*Primary template for `promise`.*
- class `promise< _Res & >`  
*Partial specialization for `promise<R&>`*
- class `promise< void >`  
*Explicit specialization for `promise<void>`*
- class `queue`

*A standard container giving FIFO behavior.*

- struct [random\\_access\\_iterator\\_tag](#)

*Random-access iterators support a superset of bidirectional /// iterator operations.*

- class [random\\_device](#)
- class [range\\_error](#)
- struct [ratio](#)

*Provides compile-time rational arithmetic.*

- struct [ratio\\_add](#)

*ratio\_add*

- struct [ratio\\_divide](#)

*ratio\_divide*

- struct [ratio\\_equal](#)

*ratio\_equal*

- struct [ratio\\_greater](#)

*ratio\_greater*

- struct [ratio\\_greater\\_equal](#)

*ratio\_greater\_equal*

- struct [ratio\\_less](#)

*ratio\_less*

- struct [ratio\\_less\\_equal](#)

*ratio\_less\_equal*

- struct [ratio\\_multiply](#)

*ratio\_multiply*

- struct [ratio\\_not\\_equal](#)

*ratio\_not\_equal*

- struct [ratio\\_subtract](#)

*ratio\_subtract*

- class [raw\\_storage\\_iterator](#)
- class [recursive\\_mutex](#)



*recursive\_mutex*

- class `recursive_timed_mutex`

*recursive\_timed\_mutex*

- class `reference_wrapper`

*Primary class template for `reference_wrapper`.*

- class `regex_error`

*A regular expression exception class.*

*The regular expression library throws objects of this class on error.*

- class `regex_iterator`

- class `regex_token_iterator`

- struct `regex_traits`

*Describes aspects of a regular expression.*

- struct `remove_reference`

*remove\_reference*

- class `reverse_iterator`

- class `runtime_error`

*One of two subclasses of exception.*

- class `seed_seq`

*The `seed_seq` class generates sequences of seeds for random number generators.*

- class `set`

*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

- class `shared_future`

*Primary template for `shared_future`.*

- class `shared_future<_Res & >`

*Partial specialization for `shared_future<R&>`*

- class `shared_future< void >`

*Explicit specialization for `shared_future<void>`*

- class `shared_ptr`

*A smart pointer with reference-counted copy semantics.*

- class [shuffle\\_order\\_engine](#)  
*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.*
- class [slice](#)  
*Class defining one-dimensional subset of an array.*
- class [slice\\_array](#)  
*Reference to one-dimensional subset of an array.*
- class [stack](#)  
*A standard container giving FILO behavior.*
- class [student\\_t\\_distribution](#)  
*A [student\\_t\\_distribution](#) random number distribution.*
- class [sub\\_match](#)
- class [system\\_error](#)  
*Thrown to indicate error code of underlying system.*
- class [thread](#)  
*thread*
- class [time\\_base](#)  
*Time format ordering data.  
This class provides an enum representing different orderings of time: day, month, and year.*
- class [time\\_get](#)  
*Primary class template [time\\_get](#).  
This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*
- class [time\\_get\\_byname](#)  
*class [time\\_get\\_byname](#) [22.2.5.2].*
- class [time\\_put](#)  
*Primary class template [time\\_put](#).  
This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*
- class [time\\_put\\_byname](#)  
*class [time\\_put\\_byname](#) [22.2.5.4].*

- class `timed_mutex`  
*timed\_mutex*
- struct `try_to_lock_t`  
*Try to acquire ownership of the mutex without blocking.*
- class `tuple`  
*tuple*
- class `tuple< _T1, _T2 >`  
*tuple (2-element), with construction and assignment from a pair.*
- struct `tuple_element< 0, tuple< _Head, _Tail...> >`
- struct `tuple_element< __i, tuple< _Head, _Tail...> >`
- struct `tuple_size< tuple< _Elements...> >`  
*class tuple\_size*
- class `type_info`  
*Part of RTTI.*
- struct `unary_function`
- class `unary_negate`  
*One of the negation functors.*
- class `underflow_error`
- class `uniform_int_distribution`  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- class `uniform_real_distribution`  
*Uniform continuous distribution for random numbers.*
- class `unique_lock`  
*unique\_lock*
- class `unique_ptr`  
*20.7.12.2 unique\_ptr for single objects.*
- class `unique_ptr< _Tp[ ], _Tp_Deleter >`  
*20.7.12.3 unique\_ptr for array objects with a runtime length*
- class `unordered_map`

*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*

- class [unordered\\_multimap](#)

*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*

- class [unordered\\_multiset](#)

*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*

- class [unordered\\_set](#)

*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*

- class [valarray](#)

*Smart array designed to support numeric processing.*

- class [vector](#)

*A standard container which offers fixed time access to individual elements in any order.*

- class [vector< bool, \\_Alloc >](#)

*A specialization of vector for booleans which offers fixed time access to individual elements in any order.*

- class [weak\\_ptr](#)

*A smart pointer with weak semantics.*

- class [weibull\\_distribution](#)

*A [weibull\\_distribution](#) random number distribution.*

## Typedefs

- typedef struct std::\_\_atomic\_flag\_base [\\_\\_atomic\\_flag\\_base](#)
- typedef FILE [\\_\\_c\\_file](#)
- typedef \_\_locale\_t [\\_\\_c\\_locale](#)
- typedef \_\_pthread\_mutex\_t [\\_\\_c\\_lock](#)
- typedef unsigned long [\\_Bit\\_type](#)
- typedef [atomic\\_short](#) [atomic\\_int\\_fast16\\_t](#)
- typedef [atomic\\_int](#) [atomic\\_int\\_fast32\\_t](#)
- typedef [atomic\\_llong](#) [atomic\\_int\\_fast64\\_t](#)

- typedef [atomic\\_schar](#) [atomic\\_int\\_fast8\\_t](#)
- typedef [atomic\\_short](#) [atomic\\_int\\_least16\\_t](#)
- typedef [atomic\\_int](#) [atomic\\_int\\_least32\\_t](#)
- typedef [atomic\\_llong](#) [atomic\\_int\\_least64\\_t](#)
- typedef [atomic\\_schar](#) [atomic\\_int\\_least8\\_t](#)
- typedef [atomic\\_llong](#) [atomic\\_intmax\\_t](#)
- typedef [atomic\\_long](#) [atomic\\_intptr\\_t](#)
- typedef [atomic\\_long](#) [atomic\\_ptrdiff\\_t](#)
- typedef [atomic\\_ulong](#) [atomic\\_size\\_t](#)
- typedef [atomic\\_long](#) [atomic\\_ssize\\_t](#)
- typedef [atomic\\_ushort](#) [atomic\\_uint\\_fast16\\_t](#)
- typedef [atomic\\_uint](#) [atomic\\_uint\\_fast32\\_t](#)
- typedef [atomic\\_ullong](#) [atomic\\_uint\\_fast64\\_t](#)
- typedef [atomic\\_uchar](#) [atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic\\_ushort](#) [atomic\\_uint\\_least16\\_t](#)
- typedef [atomic\\_uint](#) [atomic\\_uint\\_least32\\_t](#)
- typedef [atomic\\_ullong](#) [atomic\\_uint\\_least64\\_t](#)
- typedef [atomic\\_uchar](#) [atomic\\_uint\\_least8\\_t](#)
- typedef [atomic\\_ullong](#) [atomic\\_uintmax\\_t](#)
- typedef [atomic\\_ulong](#) [atomic\\_uintptr\\_t](#)
- typedef [ratio](#)< 1, 1000000000000000000 > [atto](#)
- typedef [ratio](#)< 1, 100 > [centi](#)
- typedef [match\\_results](#)< const char \* > [cmatch](#)
- typedef [regex\\_iterator](#)< const char \* > [cregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const char \* > [cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [csub\\_match](#)
- typedef [ratio](#)< 10, 1 > [deca](#)
- typedef [ratio](#)< 1, 10 > [deci](#)
- typedef [minstd\\_rand0](#) [default\\_random\\_engine](#)
- typedef [ratio](#)< 1000000000000000000, 1 > [exa](#)
- typedef [ratio](#)< 1, 1000000000000000 > [femto](#)
- typedef [basic\\_filebuf](#)< char > [filebuf](#)
- typedef [basic\\_fstream](#)< char > [fstream](#)
- typedef [ratio](#)< 1000000000, 1 > [giga](#)
- typedef [ratio](#)< 100, 1 > [hecto](#)
- typedef [basic\\_ifstream](#)< char > [ifstream](#)
- typedef [basic\\_ios](#)< char > [ios](#)
- typedef [basic\\_iostream](#)< char > [iostream](#)
- typedef [basic\\_istream](#)< char > [istream](#)
- typedef [basic\\_istreamstream](#)< char > [istreamstream](#)
- typedef [ratio](#)< 1000, 1 > [kilo](#)
- typedef [shuffle\\_order\\_engine](#)< [minstd\\_rand0](#), 256 > [knuth\\_b](#)
- typedef [ratio](#)< 1000000, 1 > [mega](#)

- typedef enum [std::memory\\_order](#) [memory\\_order](#)
- typedef [ratio](#)< 1, 1000000 > **micro**
- typedef [ratio](#)< 1, 1000 > **milli**
- typedef [linear\\_congruential\\_engine](#)< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > [minstd\\_rand](#)
- typedef [linear\\_congruential\\_engine](#)< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [minstd\\_rand0](#)
- typedef [mersenne\\_twister\\_engine](#)< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef [mersenne\\_twister\\_engine](#)< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee000000000ULL, 43, 6364136223846793005ULL > [mt19937\\_64](#)
- typedef [ratio](#)< 1, 1000000000 > **nano**
- typedef void(\* [new\\_handler](#) )()
- typedef [basic\\_ofstream](#)< char > [ofstream](#)
- typedef [basic\\_ostream](#)< char > [ostream](#)
- typedef [basic\\_ostringstream](#)< char > [ostringstream](#)
- typedef [ratio](#)< 1000000000000000, 1 > **peta**
- typedef [ratio](#)< 1, 1000000000000 > **pico**
- typedef \_\_PTRDIFF\_TYPE\_\_ [ptrdiff\\_t](#)
- typedef [discard\\_block\\_engine](#)< [ranlux24\\_base](#), 223, 23 > **ranlux24**
- typedef [subtract\\_with\\_carry\\_engine](#)< uint\_fast32\_t, 24, 10, 24 > **ranlux24\_base**
- typedef [discard\\_block\\_engine](#)< [ranlux48\\_base](#), 389, 11 > **ranlux48**
- typedef [subtract\\_with\\_carry\\_engine](#)< uint\_fast64\_t, 48, 5, 12 > **ranlux48\_base**
- typedef [basic\\_regex](#)< char > [regex](#)
- typedef \_\_SIZE\_TYPE\_\_ [size\\_t](#)
- typedef [match\\_results](#)< string::const\_iterator > **smatch**
- typedef [regex\\_iterator](#)< string::const\_iterator > **sregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< string::const\_iterator > [sregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< string::const\_iterator > [ssub\\_match](#)
- typedef [basic\\_streambuf](#)< char > [streambuf](#)
- typedef long long [streamoff](#)
- typedef [fpos](#)< mbstate\_t > [streampos](#)
- typedef [ptrdiff\\_t](#) [streamsize](#)
- typedef [basic\\_string](#)< char > [string](#)
- typedef [basic\\_stringbuf](#)< char > [stringbuf](#)
- typedef [basic\\_stringstream](#)< char > [stringstream](#)
- typedef [ratio](#)< 1000000000000, 1 > **tera**
- typedef void(\* [terminate\\_handler](#) )()

- typedef `fpos`< `mbstate_t` > `u16streampos`
- typedef `basic_string`< `char16_t` > `u16string`
- typedef `fpos`< `mbstate_t` > `u32streampos`
- typedef `basic_string`< `char32_t` > `u32string`
- typedef void(\* `unexpected_handler` )()
- typedef `match_results`< const `wchar_t` \* > `wcmatch`
- typedef `regex_iterator`< const `wchar_t` \* > `wcregex_iterator`
- typedef `regex_token_iterator`< const `wchar_t` \* > `wcregex_token_iterator`
- typedef `sub_match`< const `wchar_t` \* > `wcsub_match`
- typedef `basic_filebuf`< `wchar_t` > `wfilebuf`
- typedef `basic_fstream`< `wchar_t` > `wfstream`
- typedef `basic_ifstream`< `wchar_t` > `wifstream`
- typedef `basic_ios`< `wchar_t` > `wios`
- typedef `basic_iostream`< `wchar_t` > `wiostream`
- typedef `basic_istream`< `wchar_t` > `wistream`
- typedef `basic_istreamstream`< `wchar_t` > `wistreamstream`
- typedef `basic_ofstream`< `wchar_t` > `wofstream`
- typedef `basic_ostream`< `wchar_t` > `wostream`
- typedef `basic_ostreamstream`< `wchar_t` > `wostreamstream`
- typedef `basic_regex`< `wchar_t` > `wregex`
- typedef `match_results`< `wstring::const_iterator` > `wsmatch`
- typedef `regex_iterator`< `wstring::const_iterator` > `wsregex_iterator`
- typedef `regex_token_iterator`< `wstring::const_iterator` > `wsregex_token_iterator`
- typedef `sub_match`< `wstring::const_iterator` > `wssub_match`
- typedef `basic_streambuf`< `wchar_t` > `wstreambuf`
- typedef `fpos`< `mbstate_t` > `wstreampos`
- typedef `basic_string`< `wchar_t` > `wstring`
- typedef `basic_stringbuf`< `wchar_t` > `wstringbuf`
- typedef `basic_stringstream`< `wchar_t` > `wstringstream`

## Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }
- enum { `_S_word_bit` }
- enum `_Ios_Fmtflags` {  
`_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`,  
`_S_internal`, `_S_left`, `_S_oct`, `_S_right`,  
`_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`,  
`_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`,  
`_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end` }

- enum `_Ios_Iostate` {  
`_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
`_S_ios_iostate_end` }
- enum `_Ios_Openmode` {  
`_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
`_S_out`, `_S_trunc`, `_S_ios_openmode_end` }
- enum `_Ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }
- enum `_Manager_operation` { `__get_type_info`, `__get_functor_ptr`, `__clone_functor`, `__destroy_functor` }
- enum `_Rb_tree_color` { `_S_red`, `_S_black` }
- enum `cv_status` { `no_timeout`, `timeout` }
- enum `errc` {  
`address_family_not_supported`, `address_in_use`, `address_not_available`,  
`already_connected`,  
`argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`,  
`bad_message`, `broken_pipe`, `connection_aborted`, `connection_already_in_progress`,  
`connection_refused`, `connection_reset`, `cross_device_link`, `destination_address_required`,  
`device_or_resource_busy`, `directory_not_empty`, `executable_format_error`, `file_exists`,  
`file_too_large`, `filename_too_long`, `function_not_supported`, `host_unreachable`,  
`identifier_removed`, `illegal_byte_sequence`, `inappropriate_io_control_operation`, `interrupted`,  
`invalid_argument`, `invalid_seek`, `io_error`, `is_a_directory`,  
`message_size`, `network_down`, `network_reset`, `network_unreachable`,  
`no_buffer_space`, `no_child_process`, `no_link`, `no_lock_available`,  
`no_message_available`, `no_message`, `no_protocol_option`, `no_space_on_device`,  
`no_stream_resources`, `no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`,  
`no_such_process`, `not_a_directory`, `not_a_socket`, `not_a_stream`,  
`not_connected`, `not_enough_memory`, `not_supported`, `operation_canceled`,  
`operation_in_progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`,  
`owner_dead`, `permission_denied`, `protocol_error`, `protocol_not_supported`,



- `read_only_file_system`, `resource_deadlock_would_occur`, `resource_ - unavailable_try_again`, `result_out_of_range`,
- `state_not_recoverable`, `stream_timeout`, `text_file_busy`, `timed_out`,
- `too_many_files_open_in_system`, `too_many_files_open`, `too_many_links`, `too_many_symbolic_link_levels`,
- `value_too_large`, `wrong_protocol_type` }
- enum `float_denorm_style` { `denorm_indeterminate`, `denorm_absent`, `denorm_ - present` }
- enum `float_round_style` {  
`round_indeterminate`, `round_toward_zero`, `round_to_nearest`, `round_toward_ - infinity`,  
`round_toward_neg_infinity` }
- enum `future_errc` { `broken_promise`, `future_already_retrieved`, `promise_ - already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }
- enum `memory_order` {  
`memory_order_relaxed`, `memory_order_consume`, `memory_order_ - acquire`, `memory_order_release`,  
`memory_order_acq_rel`, `memory_order_seq_cst` }

## Functions

- `template<typename _CharT >`  
`_CharT * __add_grouping` (`_CharT * __s`, `_CharT __sep`, `const char * __gbeg`,  
`size_t __gsize`, `const _CharT * __first`, `const _CharT * __last`)
- `template<typename _Tp >`  
`_Tp * __addressof` (`_Tp & __r`)
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __adjust_heap` (`_RandomAccessIterator __first`, `_Distance __holeIndex`,  
`_Distance __len`, `_Tp __value`)
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _ - Compare >`  
`void __adjust_heap` (`_RandomAccessIterator __first`, `_Distance __holeIndex`,  
`_Distance __len`, `_Tp __value`, `_Compare __comp`)
- `template<typename _InputIterator, typename _Distance >`  
`void __advance` (`_InputIterator & __i`, `_Distance __n`, `input_iterator_tag`)
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void __advance` (`_BidirectionalIterator & __i`, `_Distance __n`, `bidirectional_ - iterator_tag`)
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __advance` (`_RandomAccessIterator & __i`, `_Distance __n`, `random_ - access_iterator_tag`)

- void **\_\_atomic\_flag\_wait\_explicit** (\_\_atomic\_flag\_base \*, [memory\\_order](#)) - GLIBCXX\_NOTHROW
- **\_\_attribute\_\_** ((\_\_pure\_\_)) [\\_Rb\\_tree\\_node\\_base](#) \*[\\_Rb\\_tree\\_increment](#)([\\_Rb\\_tree\\_node\\_base](#) \*\_\_x) throw ()
- **\_\_attribute\_\_** ((\_\_const\_\_)) [\\_\\_atomic\\_flag\\_base](#) \*[\\_\\_atomic\\_flag\\_for\\_address](#)(const void \*\_\_z) - GLIBCXX\_NOTHROW
- [memory\\_order](#) **\_\_calculate\_memory\_order** ([memory\\_order](#) \_\_m)
- template<typename \_Member, typename \_Class>  
  [\\_Mem\\_fn](#)< \_Member \_Class::\*> **\_\_callable\_functor** (\_Member \_Class::\*const &\_\_p)
- template<typename \_Functor>  
  [\\_Functor](#) & **\_\_callable\_functor** (\_Functor &\_\_f)
- template<typename \_Member, typename \_Class>  
  [\\_Mem\\_fn](#)< \_Member \_Class::\*> **\_\_callable\_functor** (\_Member \_Class::\*&\_\_p)
- template<typename \_Facet>  
  const [\\_Facet](#) & **\_\_check\_facet** (const [\\_Facet](#) \*\_\_f)
- template<typename \_RandomAccessIterator, typename \_Distance>  
  void **\_\_chunk\_insertion\_sort** (\_RandomAccessIterator \_\_first, - [RandomAccessIterator](#) \_\_last, [\\_Distance](#) \_\_chunk\_size)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Compare>  
  void **\_\_chunk\_insertion\_sort** (\_RandomAccessIterator \_\_first, - [RandomAccessIterator](#) \_\_last, [\\_Distance](#) \_\_chunk\_size, [\\_Compare](#) \_\_comp)
  
- template<typename \_Tp>  
  [\\_Tp](#) **\_\_cmath\_power** (\_Tp, unsigned int)
- template<typename \_Tp>  
  [\\_Tp](#) **\_\_complex\_abs** (const [complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [\\_Tp](#) **\_\_complex\_arg** (const [complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [complex](#)< \_Tp> **\_\_complex\_cos** (const [complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [complex](#)< \_Tp> **\_\_complex\_cosh** (const [complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [complex](#)< \_Tp> **\_\_complex\_exp** (const [complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [complex](#)< \_Tp> **\_\_complex\_log** (const [complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [complex](#)< \_Tp> **\_\_complex\_pow** (const [complex](#)< \_Tp> &\_\_x, const [complex](#)< \_Tp> &\_\_y)
- template<typename \_Tp>  
  [std::complex](#)< \_Tp> **\_\_complex\_proj** (const [std::complex](#)< \_Tp> &\_\_z)
- template<typename \_Tp>  
  [complex](#)< \_Tp> **\_\_complex\_sin** (const [complex](#)< \_Tp> &\_\_z)

- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc __attribute__((__unused__)),`  
`char *__out, const int __size __attribute__((__unused__)), const char *__fmt,...)`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (const`

```

_CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT
> > >)
• template<bool _IsMove, typename _CharT >
 __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__-
 type __copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT >
 >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)
• template<bool _IsMove, typename _II, typename _OI >
 _OI __copy_move_a2 (_II __first, _II __last, _OI __result)
• template<bool _IsMove, typename _BI1, typename _BI2 >
 _BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)
• template<bool _IsMove, typename _BI1, typename _BI2 >
 _BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)
• template<typename _InputIterator, typename _Size, typename _OutputIterator >
 _OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator _-
 _result, input_iterator_tag)
• template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >
 _OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _-
 OutputIterator __result, random_access_iterator_tag)
• template<typename _CharT, typename _Traits >
 streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin,
 basic_streambuf< _CharT, _Traits > *__sbout)
• template<>
 streamsize __copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_-
 streambuf< char > *__sbout, bool &__ineof)
• template<>
 streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin,
 basic_streambuf< wchar_t > *__sbout, bool &__ineof)
• template<typename _CharT, typename _Traits >
 streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *,
 basic_streambuf< _CharT, _Traits > *, bool &)
• size_t __deque_buf_size (size_t __size)
• template<typename _InputIterator >
 iterator_traits< _InputIterator >::difference_type __distance (_InputIterator _-
 _first, _InputIterator __last, input_iterator_tag)
• template<typename _RandomAccessIterator >
 iterator_traits< _RandomAccessIterator >::difference_type __distance (_-
 RandomAccessIterator __first, _RandomAccessIterator __last, random_access_-
 iterator_tag)
• template<typename _II1, typename _II2 >
 bool __equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)
• template<typename _ForwardIterator, typename _Tp >
 __gnu_cxx::__enable_if<!__is_scalar< _Tp >::__value, void >::__type _-
 fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

```

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type __fill_a`  
`(_Tp *__first, _Tp *__last, const _Tp &__c)`
- `void __fill_bvector (_Bit_iterator __first, _Bit_iterator __last, bool __x)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if<!__is_scalar< _Tp >::__value, _OutputIterator >::__-`  
`__type __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type __fill_-`  
`n_a (_Tp *__first, _Size __n, const _Tp &__c)`
- `template<typename _RandomAccessIterator >`  
`void __final_insertion_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __final_insertion_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator __find (_InputIterator __first, _InputIterator __last, const _Tp &_-`  
`__val, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator __find (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_-`  
`iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_-`  
`iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`  
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,`  
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_-`  
`iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BinaryPredicate >`  
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,`  
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_-`  
`iterator_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate`  
`__pred, input_iterator_tag)`

- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator __find_if (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator __find_if_not (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement __gcd (_EuclideanRingElement __m, _-`  
`EuclideanRingElement __n)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`__add_ref< _Head >::type __get_helper (_Tuple_impl< __i, _Head, _Tail...>`  
`&__t)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`__add_c_ref< _Head >::type __get_helper (const _Tuple_impl< __i, _Head,`  
`_Tail...> &__t)`
- `template<typename _Ex >`  
`const nested\_exception * __get_nested_exception (const _Ex &__ex)`
- `mutex & __get_once_mutex ()`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__middle, _RandomAccessIterator __last)`
- `template<typename _Tp >`  
`size_t __iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t`  
`*, iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t`  
`*__outbytes)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`  
`_ForwardIterator __inplace_stable_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`  
`void __inplace_stable_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __inplace_stable_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT >`  
`_GLIBCXX_END_LDBL_NAMESPACE int \_\_int\_to\_char (_CharT *__-`  
`bufend, _ValueT __v, const _CharT *__lit, ios\_base::fmtflags __flags, bool _`  
`__dec)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void \_\_introsort (_RandomAccessIterator __first, _RandomAccessIterator _`  
`__nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void \_\_introsort (_RandomAccessIterator __first, _RandomAccessIterator _`  
`__nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _Compare __comp, _Distance`  
`__n)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance \_\_is\_heap\_until (_RandomAccessIterator __first, _Distance __n, _`  
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance \_\_is\_heap\_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _Iter >`  
`iterator\_traits< _Iter >::iterator_category \_\_iterator\_category (const _Iter &)`
- `template<typename _II1, typename _II2 >`  
`bool \_\_lexicographical\_compare\_aux (_II1 __first1, _II1 __last1, _II2 __first2,`  
`__II2 __last2)`
- `template<typename _Size >`  
`_Size \_\_lg (_Size __n)`
- `int \_\_lg (int __n)`
- `long \_\_lg (long __n)`

- long long **\_\_lg** (long long \_\_n)
- template<typename \_BidirectionalIterator, typename \_Distance, typename \_Pointer >  
void **\_\_merge\_adaptive** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_middle, \_BidirectionalIterator \_\_last, \_Distance \_\_len1, \_Distance \_\_len2, \_Pointer \_\_buffer, \_Distance \_\_buffer\_size)
- template<typename \_BidirectionalIterator, typename \_Distance, typename \_Pointer, typename \_Compare >  
void **\_\_merge\_adaptive** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_middle, \_BidirectionalIterator \_\_last, \_Distance \_\_len1, \_Distance \_\_len2, \_Pointer \_\_buffer, \_Distance \_\_buffer\_size, \_Compare \_\_comp)
- template<typename \_BidirectionalIterator1, typename \_BidirectionalIterator2, typename \_BidirectionalIterator3 >  
\_BidirectionalIterator3 **\_\_merge\_backward** (\_BidirectionalIterator1 \_\_first1, \_BidirectionalIterator1 \_\_last1, \_BidirectionalIterator2 \_\_first2, \_BidirectionalIterator2 \_\_last2, \_BidirectionalIterator3 \_\_result)
- template<typename \_BidirectionalIterator1, typename \_BidirectionalIterator2, typename \_BidirectionalIterator3, typename \_Compare >  
\_BidirectionalIterator3 **\_\_merge\_backward** (\_BidirectionalIterator1 \_\_first1, \_BidirectionalIterator1 \_\_last1, \_BidirectionalIterator2 \_\_first2, \_BidirectionalIterator2 \_\_last2, \_BidirectionalIterator3 \_\_result, \_Compare \_\_comp)
- template<typename \_RandomAccessIterator1, typename \_RandomAccessIterator2, typename \_Distance >  
void **\_\_merge\_sort\_loop** (\_RandomAccessIterator1 \_\_first, \_RandomAccessIterator1 \_\_last, \_RandomAccessIterator2 \_\_result, \_Distance \_\_step\_size)
- template<typename \_RandomAccessIterator1, typename \_RandomAccessIterator2, typename \_Distance, typename \_Compare >  
void **\_\_merge\_sort\_loop** (\_RandomAccessIterator1 \_\_first, \_RandomAccessIterator1 \_\_last, \_RandomAccessIterator2 \_\_result, \_Distance \_\_step\_size, \_Compare \_\_comp)
- template<typename \_RandomAccessIterator, typename \_Pointer >  
void **\_\_merge\_sort\_with\_buffer** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Pointer \_\_buffer)
- template<typename \_RandomAccessIterator, typename \_Pointer, typename \_Compare >  
void **\_\_merge\_sort\_with\_buffer** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Pointer \_\_buffer, \_Compare \_\_comp)
- template<typename \_BidirectionalIterator, typename \_Distance >  
void **\_\_merge\_without\_buffer** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_middle, \_BidirectionalIterator \_\_last, \_Distance \_\_len1, \_Distance \_\_len2)
- template<typename \_BidirectionalIterator, typename \_Distance, typename \_Compare >  
void **\_\_merge\_without\_buffer** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_middle, \_BidirectionalIterator \_\_last, \_Distance \_\_len1, \_Distance \_\_len2, \_Compare \_\_comp)



- `template<typename _Iterator >`  
`_Miter_base< _Iterator >::iterator_type __miter_base (_Iterator __it)`
- `template<typename _Iterator >`  
`void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _Iterator, typename _Compare >`  
`void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >::iterator_type __niter_base (_Iterator __it)`
- `void __once_proxy ()`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template ostream & __ostream_insert (ostream &, const char *, streamsize)`
- `template wostream & __ostream_insert (wostream &, const wchar_t *, streamsize)`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp __pow_helper (_Tp __x, int __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`

- `template<typename _RandomAccessIterator >`  
`void \_\_reverse (_RandomAccessIterator __first, _RandomAccessIterator __last,  
random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator >`  
`void \_\_reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,  
bidirectional\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void \_\_rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last, forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator >`  
`void \_\_rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _-`  
`BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`void \_\_rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`middle, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _-`  
`Distance >`  
`\_BidirectionalIterator1 \_\_rotate\_adaptive (_BidirectionalIterator1 __first, _-`  
`BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __-`  
`len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_`  
`size)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp >`  
`\_ForwardIterator \_\_search\_n (_ForwardIterator __first, _ForwardIterator __last,`  
`_Integer __count, const _Tp &__val, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp >`  
`\_RandomAccessIter \_\_search\_n (_RandomAccessIter __first, _-`  
`RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random\_`  
`access\_iterator\_tag)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _-`  
`BinaryPredicate >`  
`\_ForwardIterator \_\_search\_n (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`  
`std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _-`  
`BinaryPredicate >`  
`\_RandomAccessIter \_\_search\_n (_RandomAccessIter __first, _-`  
`RandomAccessIter __last, _Integer __count, const _Tp &__val, _-`  
`BinaryPredicate __binary_pred, std::random\_access\_iterator\_tag)`
- `void \_\_set\_once\_functor\_lock\_ptr (unique\_lock< mutex > *)`
- `template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _-`  
`Distance >`  
`\_ForwardIterator \_\_stable\_partition\_adaptive (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer,`  
`_Distance __buffer_size)`

- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance >`  
`void __stable_sort_adaptive (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename`  
`_Compare >`  
`void __stable_sort_adaptive (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,`  
`_Compare __comp)`
- `void __throw_bad_alloc (void) __attribute__((__noreturn__))`
- `void __throw_bad_cast (void) __attribute__((__noreturn__))`
- `void __throw_bad_exception (void) __attribute__((__noreturn__))`
- `void __throw_bad_function_call () __attribute__((__noreturn__))`
- `void __throw_bad_typeid (void) __attribute__((__noreturn__))`
- `void __throw_domain_error (const char *) __attribute__((__noreturn__))`
- `void __throw_future_error (int) __attribute__((__noreturn__))`
- `void __throw_invalid_argument (const char *) __attribute__((__noreturn__))`
- `void __throw_ios_failure (const char *) __attribute__((__noreturn__))`
- `void __throw_length_error (const char *) __attribute__((__noreturn__))`
- `void __throw_logic_error (const char *) __attribute__((__noreturn__))`
- `void __throw_out_of_range (const char *) __attribute__((__noreturn__))`
- `void __throw_overflow_error (const char *) __attribute__((__noreturn__))`
- `void __throw_range_error (const char *) __attribute__((__noreturn__))`
- `void __throw_regex_error (regex_constants::error_type __ecode)`
- `void __throw_runtime_error (const char *) __attribute__((__noreturn__))`
- `void __throw_system_error (int) __attribute__((__noreturn__))`
- `void __throw_underflow_error (const char *) __attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void __throw_with_nested (_Ex &&, const nested_exception *e) __-`  
`attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void __throw_with_nested (_Ex &&...) __attribute__((__noreturn__))`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _-`  
`UIIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (const tuple< _-`  
`TElements...> &__t, const __index_holder< _TIdx...> &__, const tuple< _-`  
`UElements...> &__u, const __index_holder< _UIIdx...> &__)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _-`  
`UIIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (tuple< _-`  
`TElements...> &&__t, const __index_holder< _TIdx...> &__, const tuple< _-`  
`UElements...> &__u, const __index_holder< _UIIdx...> &__)`

- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (const tuple< _TElements...> &__t, const __index_holder< _TIdx...> &, tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (tuple< _TElements...> &&__t, const __index_holder< _TIdx...> &, tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename _RandomAccessIterator >`  
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void __unguarded_linear_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`  
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void __uninitialized_construct_buf (_ForwardIterator __first, _ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`

- 
- ```

    _ForwardIterator __uninitialized_copy_move (_InputIterator1 __first1, _-
    InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-
    ForwardIterator __result, _Allocator &__alloc)

```
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator __uninitialized_copy_n (_InputIterator __first, _Size __n, _-`
`ForwardIterator __result, input_iterator_tag)`
 - `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator __uninitialized_copy_n (_RandomAccessIterator __first, _-`
`Size __n, _ForwardIterator __result, random_access_iterator_tag)`
 - `template<typename _ForwardIterator >`
`void __uninitialized_default (_ForwardIterator __first, _ForwardIterator __-`
`last)`
 - `template<typename _ForwardIterator, typename _Allocator >`
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __-`
`last, _Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Tp >`
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __-`
`last, allocator< _Tp > &)`
 - `template<typename _ForwardIterator, typename _Size >`
`void __uninitialized_default_n (_ForwardIterator __first, _Size __n)`
 - `template<typename _ForwardIterator, typename _Size, typename _Allocator >`
`void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _-`
`Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, alloca-`
`tor< _Tp > &)`
 - `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x, _Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x, allocator< _Tp2 > &)`
 - `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _-`
`Allocator >`
`_ForwardIterator __uninitialized_fill_move (_ForwardIterator __result, _-`
`ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator`
`__last, _Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`
`&__x, _Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`
`&__x, allocator< _Tp2 > &)`
-

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_move_a (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator __uninitialized_move_copy (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _`
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _`
`Allocator >`
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator`
`&__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _`
`ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator _`
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_-`
`iterator_tag, output_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __`
`last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last,`
`_ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`
`forward_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _`
`OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`
`output_iterator_tag)`
- `template<typename _Bi_iter >`
`const sub_match< _Bi_iter > & __unmatched_sub ()`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__`
`restrict __b)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp`
`*__restrict __b)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b,`
`size_t __n, size_t __s)`

- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1,`
`_Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict`
`__i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *--`
`restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t`
`*__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array<`
`_Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n,`
`size_t __s)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array<`
`_Tp > __b, size_t __s2)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array<`
`_Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b,`
`_Array< size_t > __i)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array<`
`_Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b,`
`_Array< bool > __m)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a, size_t __s)`

- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a, _Array< bool > __m)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t >`
`__i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n,`
`_Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m,`
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n,`
`size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__-`
`__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__-`
`restrict __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i,`
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t`
`__n, _Array< _Tp > __a)`
- `template<typename _Tp >`
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i,`
`size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp`
`&__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * __valarray_get_memory (size_t __n)`
- `template<typename _Tp >`
`_Tp *__restrict__ __valarray_get_storage (size_t __n)`
- `template<typename _Ta >`
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `_GLIBCXX_PURE bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`
- `template<typename _CharT >`
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _OutIter >`
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp >`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom,`
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __-`
`_s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _-`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __-`
`m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __-`
`m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _-`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`

- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _-`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom,`
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _-`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`

- `template<typename _Tp, class _Dom>`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp`
`&__t)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _-`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, -`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, -`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, -`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _T1, typename... _Args>`
`void _Construct (_T1 *__p, _Args &&...__args)`
- `template<typename _Tp >`
`void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator`
`&__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< -`
`_Tp > &)`

- void **_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base * **_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()
- void **abort** (void) _GLIBCXX_NORETURN throw ()
- template<typename _Tp >
_Tp **abs** (const complex< _Tp > &)
- double **abs** (double __x)
- float **abs** (float __x)
- long double **abs** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **abs** (_Tp __x)
- template<class _Dom >
_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > **abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > **abs** (const valarray< _Tp > &__v)
- template<typename _InputIterator, typename _Tp >
_Tp **accumulate** (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation >
_Tp **accumulate** (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- float **acos** (float __x)
- long double **acos** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **acos** (_Tp __x)
- template<class _Dom >
_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type > **acos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > **acos** (const valarray< _Tp > &__v)
- template<typename _Tp >
_Tp * **addressof** (_Tp &__r)
- template<typename _InputIterator, typename _OutputIterator >
_OutputIterator **adjacent_difference** (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **adjacent_difference** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)

- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`
`last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance >`
`void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _Iter, typename _Predicate >`
`bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr< _Tp > allocate_shared (_Alloc __a, _Args &&...__args)`
- `template<typename _Iter, typename _Predicate >`
`bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`
`_Tp arg (const complex< _Tp > &)`
- `float asin (float __x)`
- `long double asin (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`asin (_Tp __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp`
`> &__v)`
- `template<typename _Fn, typename... _Args>`
`future< typename result_of< _Fn(_Args...)>::type > async (launch __policy,`
`_Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`
`enable_if<!is_same< typename decay< _Fn >::type, launch >::value, future<`
`decltype(std::declval< _Fn >)(std::declval< _Args >)...>>::type async (_Fn`
`&&__fn, _Args &&...__args)`
- `float atan (float __x)`
- `long double atan (long double __x)`

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`atan (_Tp __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type >`
`atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp`
`> &__v)`
- `float atan2 (float __y, float __x)`
- `long double atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_-`
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`
`>::__type, _Up >::__type atan2 (_Tp __y, _Up __x)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2`
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_-`
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _-`
`Dom, typename _Dom::value_type > &__e, const valarray< typename _-`
`Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename _Dom::value_type > atan2 (const valarray< typename _-`
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_-`
`type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _-`
`Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_-`
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `int atexit (void(*)()) throw ()`
- `bool atomic_compare_exchange_strong (atomic_bool *__a, bool *__i1, bool __i2)`
- `bool atomic_compare_exchange_strong (atomic_address *__a, void **__v1, void *__v2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)`
- `bool atomic_compare_exchange_strong_explicit (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong_explicit (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_strong_explicit (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_weak (atomic_bool *__a, bool *__i1, bool __i2)`
- `bool atomic_compare_exchange_weak (atomic_address *__a, void **__v1, void *__v2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_weak (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)`
- `bool atomic_compare_exchange_weak_explicit (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_weak_explicit (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_weak_explicit (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool atomic_exchange (atomic_bool *__a, bool __i)`
- `void * atomic_exchange (atomic_address *__a, void *__v)`
- `template<typename _ITp >`
`_ITp atomic_exchange (__atomic_base< _ITp > *__a, _ITp __i)`
- `bool atomic_exchange_explicit (atomic_bool *__a, bool __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_exchange_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void * atomic_exchange_explicit (atomic_address *__a, void *__v, memory_order __m)`
- `void * atomic_fetch_add (atomic_address *__a, ptrdiff_t __d)`

- `template<typename _ITp >`
`_ITp atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `void * atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `void atomic_flag_clear (__atomic_flag_base *__a)`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `void atomic_flag_clear_explicit (__atomic_flag_base *, memory_order) _-`
`GLIBCXX_NOTHROW`
- `bool atomic_flag_test_and_set (__atomic_flag_base *__a)`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order _-`
`m)`
- `bool atomic_flag_test_and_set_explicit (__atomic_flag_base *, memory_order _-`
`order) _GLIBCXX_NOTHROW`
- `bool atomic_is_lock_free (const atomic_bool *__a)`
- `bool atomic_is_lock_free (const atomic_address *__a)`
- `template<typename _ITp >`
`bool atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `bool atomic_load (const atomic_bool *__a)`

- void * **atomic_load** (const atomic_address * __a)
- template<typename _ITp >
_ITp **atomic_load** (const __atomic_base< _ITp > * __a)
- void * **atomic_load_explicit** (const atomic_address * __a, [memory_order](#) __m)
- bool **atomic_load_explicit** (const atomic_bool * __a, [memory_order](#) __m)
- template<typename _ITp >
_ITp **atomic_load_explicit** (const __atomic_base< _ITp > * __a, [memory_order](#) __m)
- void **atomic_store** (atomic_bool * __a, bool __i)
- void **atomic_store** (atomic_address * __a, void * __v)
- template<typename _ITp >
void **atomic_store** (__atomic_base< _ITp > * __a, _ITp __i)
- void **atomic_store_explicit** (atomic_bool * __a, bool __i, [memory_order](#) __m)
- void **atomic_store_explicit** (atomic_address * __a, void * __v, [memory_order](#) __m)
- template<typename _ITp >
void **atomic_store_explicit** (__atomic_base< _ITp > * __a, _ITp __i, [memory_order](#) __m)
- template<typename _Container >
[back_insert_iterator](#)< _Container > [back_inserter](#) (_Container & __x)
- template<typename _Filter, typename _Tp, typename _Compare >
bool **binary_search** (_Filter, _Filter, const _Tp &, _Compare)
- template<typename _Filter, typename _Tp >
bool **binary_search** (_Filter, _Filter, const _Tp &)
- template<typename _ForwardIterator, typename _Tp >
bool [binary_search](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
bool [binary_search](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)
- template<typename _Functor, typename... _ArgTypes>
_Bind< typename [_Maybe_wrap_member_pointer](#)< _Functor >::type(_ArgTypes...) > [bind](#) (_Functor __f, _ArgTypes... __args)
- template<typename _Result, typename _Functor, typename... _ArgTypes>
_Bind_result< _Result, typename [_Maybe_wrap_member_pointer](#)< _Functor >::type(_ArgTypes...) > [bind](#) (_Functor __f, _ArgTypes... __args)
- template<typename _Operation, typename _Tp >
[binder1st](#)< _Operation > [bind1st](#) (const _Operation & __fn, const _Tp & __x)
- template<typename _Operation, typename _Tp >
[binder2nd](#)< _Operation > [bind2nd](#) (const _Operation & __fn, const _Tp & __x)
- [ios_base](#) & [boolalpha](#) ([ios_base](#) & __base)
- template<typename _Callable, typename... _Args>
void [call_once](#) ([once_flag](#) & __once, _Callable __f, _Args &&... __args)
- float **ceil** (float __x)

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil`
`(_Tp __x)`
- `long double ceil (long double __x)`
- `template<typename _Tp >`
`complex< _Tp > conj (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type conj (_Tp __x)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`
- `template<typename _Iter, typename _OIter >`
`_OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _II, typename _OI >`
`_OI copy (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BIter1, typename _BIter2 >`
`_BIter2 copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Ex >`
`exception_ptr copy_exception (_Ex __ex) throw ()`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter copy_if (_Iter, _Iter, _OIter, _Predicate)`

- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter copy_n (_Iter, _Size, _OIter)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __-`
`result)`
- `float cos (float __x)`
- `long double cos (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cos`
`(_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > cos (const complex< _Tp > &)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > cos`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp`
`> &__v)`
- `float cosh (float __x)`
- `long double cosh (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`cosh (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >`
`cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type count (_InputIterator __first,`
`_InputIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __-`
`first, _InputIterator __last, _Predicate __pred)`

- exception_ptr [current_exception](#) () throw ()
- ios_base & [dec](#) (ios_base &__base)
- typedef **decltype** (nullptr) nullptr_t
- template<typename _Tp >
[add_rvalue_reference](#)< _Tp >::type **declval** () noexcept
- template<typename _InputIterator >
[iterator_traits](#)< _InputIterator >::difference_type [distance](#) (_InputIterator __-
first, _InputIterator __last)
- template<typename _Tp, typename _Tp1 >
[shared_ptr](#)< _Tp > **dynamic_pointer_cast** (const [shared_ptr](#)< _Tp1 > &__r)
- template<typename _CharT, typename _Traits >
[basic_ostream](#)< _CharT, _Traits > & [endl](#) ([basic_ostream](#)< _CharT, _Traits >
&__os)
- template<typename _CharT, typename _Traits >
[basic_ostream](#)< _CharT, _Traits > & [ends](#) ([basic_ostream](#)< _CharT, _Traits >
&__os)
- template<typename _Iter1, typename _Iter2 >
bool equal (_Iter1, _Iter1, _Iter2)
- template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >
bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate
__binary_pred)
- template<typename _II1, typename _II2 >
bool equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _FIter, typename _Tp >
[pair](#)< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare >
[pair](#)< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
[pair](#)< _ForwardIterator, _ForwardIterator > [equal_range](#) (_ForwardIterator __-
first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _ForwardIterator, typename _Tp >
[pair](#)< _ForwardIterator, _ForwardIterator > [equal_range](#) (_ForwardIterator __-
first, _ForwardIterator __last, const _Tp &__val)
- void **exit** (int) _GLIBCXX_NORETURN throw ()
- float **exp** (float __x)
- long double **exp** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **exp**
(_Tp __x)
- template<typename _Tp >
[complex](#)< _Tp > **exp** (const [complex](#)< _Tp > &)
- template<class _Dom >
_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > **exp**
(const _Expr< _Dom, typename _Dom::value_type > &__e)

- `template<typename _Tp >`
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp > &__v)`
- `float fabs (float __x)`
- `long double fabs (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`fabs (_Tp __x)`
- `template<typename _Tp >`
`void fill (const Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Filter, typename _Tp >`
`void fill (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)`
- `template<typename _OIter, typename _Size, typename _Tp >`
`_OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`

- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _-`
`ForwardIterator __first2, ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _-`
`ForwardIterator __first2, ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate _-`
`__pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `ios_base & fixed (ios_base & __base)`
- `float floor (float __x)`
- `long double floor (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`floor (_Tp __x)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits >`
`& __os)`
- `float fmod (float __x, float __y)`
- `long double fmod (long double __x, long double __y)`
- `template<typename _Iter, typename _Funct >`
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp >`
`enable_if<!is_lvalue_reference< _Tp >::__value, _Tp && >::__type forward (type-`
`name std::identity< _Tp >::type & __t)`
- `template<typename _Tp >`
`enable_if<!is_lvalue_reference< _Tp >::__value, _Tp && >::__type forward (type-`
`name std::identity< _Tp >::type && __t)`
- `template<typename _Tp >`
`enable_if< is_lvalue_reference< _Tp >::__value, _Tp >::__type forward (type-`
`name std::identity< _Tp >::type __t)`
- `template<typename _Tp >`
`enable_if< is_lvalue_reference< _Tp >::__value, _Tp >::__type forward (type-`
`name std::remove_reference< _Tp >::type && __t)`
- `float frexp (float __x, int * __exp)`

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type` **frexp** (`_Tp __x`, `int *__exp`)
- `long double frexp` (`long double __x`, `int *__exp`)
- `template<typename _Container >`
`front_insert_iterator< _Container > front_inserter` (`_Container &__x`)
- `template<typename _Filter, typename _Generator >`
`void generate` (`_Filter, _Filter, _Generator`)
- `template<typename _ForwardIterator, typename _Generator >`
`void generate` (`_ForwardIterator __first, _ForwardIterator __last, _Generator __gen`)
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`
`_RealType generate_canonical` (`_UniformRandomNumberGenerator &__g`)
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n` (`_OIter, _Size, _Generator`)
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n` (`_OutputIterator __first, _Size __n, _Generator __gen`)
- `_GLIBCXX_CONST` `const error_category & generic_category` () `throw ()`
- `template<std::size_t __i, typename... _Elements>`
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type`
`>::type get` (`tuple< _Elements...> &__t`)
- `template<std::size_t __i, typename... _Elements>`
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type`
`>::type get` (`const tuple< _Elements...> &__t`)
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * get_deleter` (`const __shared_ptr< _Tp, _Lp > &__p`)
- `template<typename _MoneyT >`
`_Get_money< _MoneyT > get_money` (`_MoneyT &__mon`, `bool __intl=false`)
- `template<typename _Tp >`
`pair< _Tp *, ptrdiff_t > get_temporary_buffer` (`ptrdiff_t __len`)
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline` (`basic_istream< _CharT, _Traits >`
`&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim`)
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline` (`basic_istream< _CharT, _Traits >`
`&__is, basic_string< _CharT, _Traits, _Alloc > &__str`)
- `template<>`
`basic_istream< char > & getline` (`basic_istream< char > &__in, basic_string<`
`char > &__str, char __delim`)
- `template<>`
`basic_istream< wchar_t > & getline` (`basic_istream< wchar_t > &__in, basic_-`
`string< wchar_t > &__str, wchar_t __delim`)

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits >`
`&__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str,`
`_CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits >`
`&__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _Facet >`
`bool has_facet (const locale &__loc) throw ()`
- `ios_base & hex (ios_base &__base)`
- `template<typename _Tp >`
`_Tp imag (const complex< _Tp > &__z)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
`__first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
`__first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _-`
`BinaryOperation1, typename _BinaryOperation2 >`
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _-`
`BinaryOperation2 __binary_op2)`
- `template<typename _BIter, typename _Compare >`
`void inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BIter >`
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > inserter (_Container &__x, _Iterator __i)`
- `ios_base & internal (ios_base &__base)`

- `template<typename _ForwardIterator, typename _Tp >`
`void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`Compare __comp)`
- `template<typename _RAIter >`
`bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _Iter, typename _Predicate >`
`bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __-`
`pred)`
- `template<typename _FIter >`
`bool is_sorted (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`
`bool is_sorted (_FIter, _FIter, _Compare)`
- `template<typename _ForwardIterator >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __-`
`comp)`
- `template<typename _FIter, typename _Compare >`
`_FIter is_sorted_until (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`
`_FIter is_sorted_until (_FIter, _FIter)`
- `template<typename _ForwardIterator >`
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __-`
`last, _Compare __comp)`

- `template<typename _CharT >`
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _FIter1, typename _FIter2 >`
`void iter_swap (_FIter1, _FIter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Tp >`
`_Tp kill_dependency (_Tp __y)`
- `float ldexp (float __x, int __exp)`
- `long double ldexp (long double __x, int __exp)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `template<typename _IIter1, typename _IIter2 >`
`bool lexicographical_compare (_IIter1, _IIter1, _IIter2, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _Compare >`
`bool lexicographical_compare (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)`
- `template<typename _II1, typename _II2 >`
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`

- `template<typename _L1, typename _L2, typename... _L3>`
`void lock (_L1 &, _L2 &, _L3 &...)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > log`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`complex< _Tp > log (const complex< _Tp > &)`
- `float log (float __x)`
- `long double log (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log`
`(_Tp __x)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp`
`> &__v)`
- `template<typename _Tp >`
`complex< _Tp > log10 (const complex< _Tp > &)`
- `long double log10 (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`log10 (_Tp __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`
`log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > log10 (const valarray<`
`_Tp > &__v)`
- `float log10 (float __x)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val)`
- `error_code make_error_code (errc __e)`
- `error_code make_error_code (future_errc __errc)`
- `error_condition make_error_condition (errc __e)`
- `error_condition make_error_condition (future_errc __errc)`
- `template<typename _Ex >`
`exception_ptr make_exception_ptr (_Ex __ex) throw ()`

- `template<typename _RandomAccessIterator >`
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`
`void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void make_heap (_RAIter, _RAIter)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > make_move_iterator (const _Iterator &__i)`
- `template<class _T1, class _T2 >`
`pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_-`
`strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > make_shared (_Args &&...__args)`
- `template<typename... _Elements>`
`tuple< typename __decay_and_strip< _Elements >::__type...> make_tuple`
`(_Elements &&...__args)`
- `template<typename _Tp >`
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`_Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp max (initializer_list< _Tp >)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __-`
`last, _Compare __comp)`
- `template<typename _Tp, typename _Class >`
`_Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::*__pm)`
- `template<typename _Ret, typename _Tp >`
`mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::*__f)(_Arg))`

- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `void * memchr (void *__s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __-`
`last, _Compare __comp)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b,`
`_Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`

- `template<typename _Filter >`
`pair< _Filter, _Filter > minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`pair< _Filter, _Filter > minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `float modf (float __x, float * __iptr)`
- `long double modf (long double __x, long double * __iptr)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`std::remove_reference< _Tp >::type && move (_Tp && __t)`
- `template<typename _II, typename _OI >`
`_OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`

- `template<typename _ForwardIterator >`
`__gnu_cxx::__enable_if< __is_iterator< _ForwardIterator >::__value, _-`
`ForwardIterator >::__type next (_ForwardIterator __x, typename iterator_`
`traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BIter >`
`bool next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __-`
`last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __-`
`last, _Compare __comp)`
- `ios_base & nboolalpha (ios_base &__base)`
- `template<typename _Iter, typename _Predicate >`
`bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`
`_Tp norm (const complex< _Tp > &)`
- `ios_base & noshowbase (ios_base &__base)`
- `ios_base & noshowpoint (ios_base &__base)`
- `ios_base & noshowpos (ios_base &__base)`
- `ios_base & noskipws (ios_base &__base)`
- `template<typename _Predicate >`
`unary_negate< _Predicate > not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > not2 (const _Predicate &__pred)`
- `ios_base & nounitbuf (ios_base &__base)`
- `ios_base & nouppercase (ios_base &__base)`
- `template<typename _RAIter >`
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-`
`nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-`
`nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base &__base)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`

- `template<typename _Iterator >`
`bool operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Tp >`
`bool operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `template<typename _Val >`
`bool operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _IntType >`
`bool operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `bool operator!= (thread::id __x, thread::id __y)`
- `template<typename _RealType >`
`bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<class _T1, class _T2 >`
`bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<typename _Val >`
`bool operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_iterator< _Val > &__y)`
- `template<typename _IntType >`
`bool operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _IntType >`
`bool operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `bool operator!= (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _StateT >`
`bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__`
`rhs)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>`
`bool operator!= (const std::linear_congruential_engine< _UIntType, __a, __c,`
`__m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c,`
`__m > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _`
`_UIntType __f>`
`bool operator!= (const std::mersenne_twister_engine< _UIntType, __w, __`
`n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`
`bool operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __`
`s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s,`
`__r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool operator!= (const std::discard_block_engine< _RandomNumberEngine, _`
`_p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine,`
`__p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool operator!= (const std::independent_bits_engine< _RandomNumberEngine,`
`__w, _UIntType > &__lhs, const std::independent_bits_engine< _`
`RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`
`bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, _`
`_k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k`
`> &__rhs)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`
`operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, type-`
`name _Dom::value_type >::result_type > operator!= (const typename`
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__v)`
- `template<typename _IntType >`
`bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const`
`std::uniform_int_distribution< _IntType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`
`_Dom::value_type >::result_type > operator!= (const valarray< typename`
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`
`> &__e)`
- `template<typename _IntType >`
`bool operator!= (const std::uniform_real_distribution< _IntType > &__d1,`
`const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::normal_distribution< _RealType > &__d1, const`
`std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const`
`std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::gamma_distribution< _RealType > &__d1, const`
`std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::fisher_f_distribution< _RealType > &__d1, const`
`std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::student_t_distribution< _RealType > &__d1, const`
`std::student_t_distribution< _RealType > &__d2)`
- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_-`
`distribution &__d2)`
- `bool operator!= (const error_code &__lhs, const error_code &__rhs)`
- `bool operator!= (const error_code &__lhs, const error_condition &__rhs)`
- `template<typename _RealType >`
`bool operator!= (const std::exponential_distribution< _RealType > &__d1,`
`const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::chi_squared_distribution< _RealType > &__d1,`
`const std::chi_squared_distribution< _RealType > &__d2)`

- `template<typename _RealType >`
`bool operator!= (const std::cauchy_distribution< _RealType > &__d1, const`
`std::cauchy_distribution< _RealType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`
`_Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type >`
`&__v)`
- `template<typename _CharT, typename _Traits >`
`bool operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const`
`istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _IntType >`
`bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1,`
`const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x,`
`const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_set< _Value, _Hash, _Pred, _Alloc, __-`
`cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__y)`
- `template<typename _RealType >`
`bool operator!= (const std::extreme_value_distribution< _RealType > &__d1,`
`const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _Bi_iter >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `bool operator!= (const error_condition &__lhs, const error_code &__rhs)`
- `template<typename _RealType >`
`bool operator!= (const std::piecewise_linear_distribution< _RealType > &__-`
`d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _Bi_iter, class _Allocator >`
`bool operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const`
`match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`
- `template<typename _BiIter >`
`bool operator!= (const sub_match< _BiIter > &__lhs, const sub_match< _-`
`BiIter > &__rhs)`
- `template<typename _RealType >`
`bool operator!= (const std::weibull_distribution< _RealType > &__d1, const`
`std::weibull_distribution< _RealType > &__d2)`

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _Res, typename... _Args>`
`bool operator!= (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>`
`bool operator!= (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator!= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_`
`ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _T1, typename _T2 >`
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const _Tp &-`
`_t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator!= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator!= (const tuple< _TElements...> &__t, const tuple< _`
`UElements...> &__u)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &-`
`_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`
`Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _IntType >`
`bool operator!= (const std::geometric_distribution< _IntType > &__d1, const`
`std::geometric_distribution< _IntType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _-`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const`
`_Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`_fun< __modulus, typename _Dom1::value_type >::result_type > operator%`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_-`
`type >::result_type > operator% (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`
`>::result_type > operator% (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_-`
`type >::result_type > operator% (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`_v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`
`>::result_type > operator% (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `_Ios_Iostate operator& (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Fmtflags operator& (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Openmode operator& (_Ios_Openmode __a, _Ios_Openmode __b)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`
`__fun< __bitwise_and, typename _Dom1::value_type >::result_type > opera-`
`tor& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _`
`Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`
`Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type >`
`&__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`
`Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`
`>::result_type > operator& (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`
`>::result_type > operator& (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const _Tp &_`
`_t, const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`
`__fun< __logical_and, typename _Dom1::value_type >::result_type > opera-`
`tor&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`
`type >::result_type > operator&& (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`
`>::result_type > operator&& (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`
`type >::result_type > operator&& (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`_v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`
`>::result_type > operator&& (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const _Tp`
`&__t, const valarray< _Tp > &__v)`

- `_Ios_Fmtflags & operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Openmode & operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-
name __fun< __multiplies, typename _Dom1::value_type >::result_type >
operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const
_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _-
Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_
type >::result_type > operator* (const _Expr< _Dom, typename _-
Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _-
Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_
type >::result_type > operator* (const _Expr< _Dom, typename _-
Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_
_v)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_
type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type
>::result_type > operator* (const typename _Dom::value_type &__t, const _-
Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_
type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type
>::result_type > operator* (const valarray< typename _Dom::value_type >
&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >
_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename
__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >
_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >
&__v, const _Tp &__t)`
- `template<typename _Tp >
_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __multiplies, _Tp >::result_type > operator* (const _Tp &__t, const
valarray< _Tp > &__v)`
- `template<typename _Tp, typename _Ref, typename _Ptr >
_Deque_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const _Deque_-
iterator< _Tp, _Ref, _Ptr > &__x)`

- `template<typename _Iterator >`
`reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator`
`>::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator`
`>::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const`
`typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const`
`valarray< typename _Dom::value_type > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_`
`string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom,`
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> operator+ (const valarray< typename _Dom::value_type > &__v, const _`
`Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`fun< __plus, typename _Dom1::value_type >::result_type > operator+ (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v,`
`const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __-`
`fun< __plus, _Tp >::result_type > operator+ (const _Tp &__t, const valarray<`
`_Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _-`
`CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc`
`> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _-`
`CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _-`
`CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`Deque_iterator< _Tp, _Ref, _Ptr >::difference_type operator- (const _-`
`Deque_iterator< _Tp, _Ref, _Ptr > &__x, const Deque_iterator< _Tp, _Ref,`
`_Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`Deque_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const _-`
`Deque_iterator< _Tp, _RefL, _PtrL > &__x, const Deque_iterator< _Tp, _-`
`RefR, _PtrR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator<`
`_IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __-`
`fun< __minus, typename _Dom1::value_type >::result_type > operator- (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_type`
`>::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type`
`> &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`
`type > operator- (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`
`type > operator- (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_type`
`>::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type`
`> &__v, const typename _Dom::value_type &__t)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::difference_type operator- (const reverse_`
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base`
`&__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`fun< __minus, _Tp >::result_type > operator- (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`fun< __minus, _Tp >::result_type > operator- (const valarray< _Tp > &__v,`
`const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __minus, _Tp >::result_type > operator- (const _Tp &__t, const valar-`
`ray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_`
`type > operator/ (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_type`
`>::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type`
`> &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_`
`type > operator/ (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_type`
`>::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type`
`> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename`
`__fun< __divides, typename _Dom1::value_type >::result_type > operator/`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &__`
`_v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &__`
`_v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > operator/ (const _Tp &__t, const valar`
`ray< _Tp > &__v)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator< (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _Bi_iter >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Iterator >`
`bool operator< (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator< (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator< (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`

- `template<typename _Tp, typename _Alloc >`
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`
`_Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multi-`
`set< _Key, _Compare, _Alloc > &__y)`
- `template<class _T1, class _T2 >`
`bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`
`_Compare, _Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _-`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> operator< (const typename _Dom::value_type &__t, const _Expr< _Dom,`
`typename _Dom::value_type > &__v)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &_-`
`__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `bool operator< (const error_code &__lhs, const error_code &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value -`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type >`
`operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const`
`typename _Dom::value_type &__t)`

- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type >`
`operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const`
`valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> operator< (const valarray< typename _Dom::value_type > &__v, const _`
`Expr< _Dom, typename _Dom::value_type > &__e)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __`
`fun< __less, typename _Dom1::value_type >::result_type > operator< (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator< (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`
`> &__y)`
- `template<typename _Bi_iter >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`
- `template<typename _Tp, typename _Seq >`
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _BiIter >`
`bool operator< (const sub_match< _BiIter > &__lhs, const sub_match< _`
`BiIter > &__rhs)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator< (const tuple< _TElements...> &__t, const tuple< _`
`UElements...> &__u)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`

- `template<typename _Tp, typename _Seq >`
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator< (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_-`
`ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_-`
`_rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _-`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _-`
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`
`const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`fun< __less, _Tp >::result_type > operator< (const _Tp &__t, const valarray<`
`_Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _-`
`Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _-`
`_Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setiosflags __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > &__os, const negative_binomial_distribution< _IntType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)`

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setprecision __f)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine,`
`__k > &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_-`
`type >::result_type > operator<< (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > operator<< (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_-`
`type >::result_type > operator<< (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-`
`__v)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setbase __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, const error_code &__e)`
- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _-`
`Traits > &__os, const _Tp &__x)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`
`operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > operator<< (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename,`
`typename > class _Base>`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _-`
`Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _-`
`Base > &__str)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_-`
`type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`

- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`

```

std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<
_CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a,
__c, __m > &__lcr)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _
_CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
• template<typename _Tp, typename _CharT, class _Traits >
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _
Traits > &__os, const complex< _Tp > &__x)
• template<typename _Tp >
_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename
__fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp
> &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp
> &__v, const _Tp &__t)
• template<typename _Tp >
_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __shift_left, _Tp >::result_type > operator<< (const _Tp &__t, const
valarray< _Tp > &__v)
• template<class _CharT, class _Traits >
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,
_Traits > &__out, thread::id __id)
• template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,
_Traits > &__os, _Resetiosflags __f)
• template<typename _Tp >
_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __less_equal, _Tp >::result_type > operator<= (const _Tp &__t, const
valarray< _Tp > &__v)
• template<typename _Tp, typename _Ref, typename _Ptr >
bool operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _
Deque_iterator< _Tp, _Ref, _Ptr > &__y)
• template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR
>
bool operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const
_Deque_iterator< _Tp, _RefR, _PtrR > &__y)
• template<typename _Tp >
_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp
> &__v, const _Tp &__t)
• template<typename _Tp, typename _Alloc >
bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc
> &__y)

```

- `template<typename _Iterator >`
`bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<class _T1, class _T2 >`
`bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator<= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _-`
`Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_-`
`type >::result_type > operator<= (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __less_equal, typename _Dom1::value_type >::result_type >`
`operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator<= (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_-`
`type >::result_type > operator<= (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`_v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > operator<= (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const *__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _BIter >`
`bool operator<= (const sub_match< _BIter > &__lhs, const sub_match< _-`
`BIter > &__rhs)`
- `bool operator<= (thread::id __x, thread::id __y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_-`
`list< _Tp, _Alloc > &__ly)`

- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator<= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > operator<= (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`
`>::result_type > operator== (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`
`_Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Iterator >`
`bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_-`
`iterator< _Iterator > &__y)`

- `template<typename _Val >`
`bool operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_`
`const_iterator< _Val > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_`
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator== (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _RealType >`
`bool operator== (const std::piecewise_constant_distribution< _RealType > &_`
`_d1, const std::piecewise_constant_distribution< _RealType > &_d2)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc, __`
`cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _`
`Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type op-`
`erator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT`
`> &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _`
`Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`
`Alloc > &__rhs)`
- `template<typename _RealType >`
`bool operator== (const std::extreme_value_distribution< _RealType > &_d1,`
`const std::extreme_value_distribution< _RealType > &_d2)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`_fun< __equal_to, typename _Dom1::value_type >::result_type > operator==`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`
`code>`
`bool operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc,`
`__cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _`
`Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _IntType >`
`bool operator== (const std::uniform_int_distribution< _IntType > &__d1, const`
`std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator== (const Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _BiIter >`
`bool operator== (const sub_match< _BiIter > &__lhs, const sub_match< _`
`BiIter > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `bool operator== (const error_code &__lhs, const error_code &__rhs)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x,`
`const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)`
- `template<typename _StateT >`
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__`
`rhs)`
- `bool operator== (const std::bernoulli_distribution &__d1, const std::bernoulli_`
`distribution &__d2)`
- `template<typename _RealType >`
`bool operator== (const std::exponential_distribution< _RealType > &__d1,`
`const std::exponential_distribution< _RealType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_`

```

type >::result_type > operator== (const _Expr< _Dom, typename _-
Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_
_v)
• template<typename _RealType >
  bool operator== (const std::weibull\_distribution< _RealType > &__d1, const
std::weibull\_distribution< _RealType > &__d2)
• template<typename _IntType >
  bool operator== (const std::uniform\_real\_distribution< _IntType > &__d1,
  const std::uniform\_real\_distribution< _IntType > &__d2)
• template<typename _RealType >
  bool operator== (const std::cauchy\_distribution< _RealType > &__d1, const
std::cauchy\_distribution< _RealType > &__d2)
• template<typename _CharT, typename _Traits >
  bool operator== (const istreambuf\_iterator< _CharT, _Traits > &__a, const
istreambuf\_iterator< _CharT, _Traits > &__b)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_
code>
  bool operator== (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _-
Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp,
_Hash, _Pred, _Alloc, __cache_hash_code > &__y)
• template<class _Dom >
  _Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _-
Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_-
type >::result_type > operator== (const _Expr< _Dom, typename _-
Dom::value_type > &__v, const typename _Dom::value_type &__t)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
  bool operator== (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >
&__x, const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
  bool operator== (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _-
Alloc > &__x, const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
> &__y)
• template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
  bool operator== (const __unordered_multiset< _Value, _Hash, _Pred, _Alloc,
__cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _-
Pred, _Alloc, __cache_hash_code > &__y)
• template<typename _Key, typename _Compare, typename _Alloc >
  bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const mul-
tiset< _Key, _Compare, _Alloc > &__y)
• template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >
  bool operator== (const unique\_ptr< _Tp, _Tp_Deleter > &__x, const unique\_-
ptr< _Up, _Up_Deleter > &__y)
• template<typename _Tp, typename _Alloc >
  bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc
> &__y)

```


- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`
`>::result_type > operator== (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _IntType >`
`bool operator== (const std::discrete_distribution< _IntType > &__d1, const`
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > operator== (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool operator== (const match_results< _Bi_iter, _Allocator > &__m1, const`
`match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`
- `template<typename _RealType >`
`bool operator== (const std::piecewise_linear_distribution< _RealType > &__`
`d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator== (const tuple< _TElements...> &__t, const tuple< _`
`UElements...> &__u)`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _RealType >`
`bool operator== (const std::normal_distribution< _RealType > &__d1, const`
`std::normal_distribution< _RealType > &__d2)`

- `template<typename _Res , typename... _Args>`
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res , typename... _Args>`
`bool operator== (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<class _T1 , class _T2 >`
`bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _Val >`
`bool operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`bool operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _CharT , typename _Traits , typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _IntType >`
`bool operator== (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _Tp , typename _Ref , typename _Ptr >`
`bool operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `bool operator> (thread::id __x, thread::id __y)`
- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`
`bool operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Iterator >`
`bool operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > operator> (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`
`_Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<class _T1, class _T2 >`
`bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator> (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`_fun< __greater, typename _Dom1::value_type >::result_type > operator>`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multi-`
`set< _Key, _Compare, _Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > operator> (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __greater, typename _Dom::value_type`
`>::result_type > operator> (const _Expr< _Dom, typename _Dom::value_`
`type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _BiIter >`
`bool operator> (const sub_match< _BiIter > &__lhs, const sub_match< _`
`BiIter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_`
`ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator> (const tuple< _TElements...> &__t, const tuple< _`
`UElements...> &__u)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`
`_Compare, _Alloc > &__y)`

- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __greater, typename _Dom::value_type`
`>::result_type > operator> (const _Expr< _Dom, typename _Dom::value_`
`type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator> (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename __-`
`fun< __greater, _Tp >::result_type > operator> (const _Tp &__t, const valar-`
`ray< _Tp > &__v)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__-`
`__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`
`> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _-`
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _BiIter >`
`bool operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _-`
`BiIter > &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _-`
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const *__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > operator>= (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > operator>= (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const mul-`
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _-`
`Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Iterator >`
`bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_-`
`iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<class _T1, class _T2 >`
`bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator>= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `bool operator>= (thread::id __x, thread::id __y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator>= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename`

```

_Dom::value_type >, typename __fun< __greater_equal, typename _-
Dom::value_type >::result_type > operator>= (const _Expr< _Dom, type-
name _Dom::value_type > &__e, const valarray< typename _Dom::value_type
> &__v)
• template<class _Dom >
  _Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename
_Dom::value_type >, typename __fun< __greater_equal, typename _-
Dom::value_type >::result_type > operator>= (const _Expr< _Dom, type-
name _Dom::value_type > &__v, const typename _Dom::value_type &__t)
• template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
  bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,
  const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
• template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR
>
  bool operator>= (const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const
Deque\_iterator< _Tp, _RefR, _PtrR > &__y)
• template<typename _IteratorL, typename _IteratorR >
  bool operator>= (const move\_iterator< _IteratorL > &__x, const move\_-
iterator< _IteratorR > &__y)
• template<typename _Tp, typename _Ref, typename _Ptr >
  bool operator>= (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const \_-
Deque\_iterator< _Tp, _Ref, _Ptr > &__y)
• template<typename _Bi_iter >
  bool operator>= (const sub\_match< _Bi_iter > &__lhs, typename iterator\_-
traits< _Bi_iter >::value_type const *__rhs)
• template<typename _Tp, typename _Alloc >
  bool operator>= (const forward\_list< _Tp, _Alloc > &__lx, const forward\_-
list< _Tp, _Alloc > &__ly)
• template<typename _CharT, typename _Traits >
  basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _-
Traits > &__is, _Resetiosflags __f)
• template<typename _RealType, typename _CharT, typename _Traits >
  std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _-
CharT, _Traits > &, std::exponential\_distribution< _RealType > &)
• template<typename _RealType, typename _CharT, typename _Traits >
  std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _-
CharT, _Traits > &, std::extreme\_value\_distribution< _RealType > &)
• template<typename _RealType, typename _CharT, typename _Traits >
  std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _-
CharT, _Traits > &__is, gamma\_distribution< _RealType > &__x)
• template<typename _RealType, typename _CharT, typename _Traits >
  std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _-
CharT, _Traits > &, std::weibull\_distribution< _RealType > &)
• template<typename _Tp, typename _CharT, class _Traits >
  basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _-
Traits > &__is, complex< _Tp > &__x)

```


- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, Setprecision __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, Setiosflags __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`
`UIntType __f, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m,`
`__r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, Get_money< _MoneyT > __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`
`operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`
`&__str)`

- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &&__is, _Tp &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &&__is, negative_binomial_distribution< _IntType > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_-`
`type >::result_type > operator>> (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &&__is, discard_block_engine< _RandomNumberEngine, __p,`
`__r > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`
`>::result_type > operator>> (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_-`
`type >::result_type > operator>> (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`_v)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &&__is, discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT`
`, typename _Traits >`

- `std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, linear_congruential_engine<_UIntType, __a, __c, __m> &__lcr)`
- `template<class _Dom>
_Expr<_BinClos<__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename __fun<__shift_right, typename _Dom::value_type>::result_type> & operator>> (const valarray<typename _Dom::value_type> &__v, const _Expr<_Dom, typename _Dom::value_type> &__e)`
 - `template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_int_distribution<_IntType> &)`
 - `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, shuffle_order_engine<_RandomNumberEngine, __k> &__x)`
 - `template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, poisson_distribution<_IntType> &__x)`
 - `template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & operator>> (basic_istream<_CharT, _Traits> &__is, _Setbase __f)`
 - `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, subtract_with_carry_engine<_UIntType, __w, __s, __r> &__x)`
 - `template<typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, std::bernoulli_distribution &__x)`
 - `template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, lognormal_distribution<_RealType> &__x)`
 - `template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, chi_squared_distribution<_RealType> &__x)`
 - `template<typename _Tp>
_Expr<_BinClos<__shift_right, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__shift_right, _Tp>::result_type> & operator>> (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
 - `template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, fisher_f_distribution<_RealType> &__x)`
 - `template<typename _Tp>
_Expr<_BinClos<__shift_right, _ValArray, _Constant, _Tp, _Tp>, typename`

```

__fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp
> &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, type-
  name __fun< __shift_right, _Tp >::result_type > operator>> (const _Tp &_
  __t, const valarray< _Tp > &__v)
• template<>
  basic_istream< char > & operator>> (basic_istream< char > &__is, basic_
  string< char > &__str)
• template<typename _CharT, typename _Traits >
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _
  Traits > &__is, _Setw __f)
• _Ios_Openmode operator^ (_Ios_Openmode __a, _Ios_Openmode __b)
• template<class _Dom >
  _Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_
  type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type
  >::result_type > operator^ (const valarray< typename _Dom::value_type >
  &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
• _Ios_Iostate operator^ (_Ios_Iostate __a, _Ios_Iostate __b)
• template<class _Dom1, class _Dom2 >
  _Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, type-
  name __fun< __bitwise_xor, typename _Dom1::value_type >::result_type >
  operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const
  _Expr< _Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
  _Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_
  type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type
  >::result_type > operator^ (const typename _Dom::value_type &__t, const _
  Expr< _Dom, typename _Dom::value_type > &__v)
• _Ios_Fmtflags operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
• template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-
  name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const _Tp &__t,
  const valarray< _Tp > &__v)
• template<class _Dom >
  _Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _
  Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_
  type >::result_type > operator^ (const _Expr< _Dom, typename _
  Dom::value_type > &__v, const typename _Dom::value_type &__t)
• template<class _Dom >
  _Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _
  Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_
  type >::result_type > operator^ (const _Expr< _Dom, typename _
  Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_
  __v)

```

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `_Ios_Openmode & operator^= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & operator^= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & operator^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > operator| (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `_Ios_Fmtflags operator| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`
`>::result_type > operator| (const typename _Dom::value_type &__t, const _-`
`Expr< _Dom, typename _Dom::value_type > &__v)`
- `_Ios_Openmode operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_-`
`type >::result_type > operator| (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`
`>::result_type > operator| (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_-`
`type >::result_type > operator| (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __bitwise_or, typename _Dom1::value_type >::result_type >`
`operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`

```

__fun< __bitwise_or, _Tp >::result_type > operator| (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  __Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename
  __fun< __bitwise_or, _Tp >::result_type > operator| (const _Tp &__t, const
valarray< _Tp > &__v)
• _Ios_Iostate operator| (_Ios_Iostate __a, _Ios_Iostate __b)
• _Ios_Openmode & operator|= (_Ios_Openmode &__a, _Ios_Openmode __b)
• _Ios_Fmtflags & operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
• _Ios_Iostate & operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)
• template<class _Dom1, class _Dom2 >
  __Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-
  name __fun< __logical_or, typename _Dom1::value_type >::result_type >
operator|| (const __Expr< _Dom1, typename _Dom1::value_type > &__v, const
  __Expr< _Dom2, typename _Dom2::value_type > &__w)
• template<typename _Tp >
  __Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename
  __fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >
  &__v, const valarray< _Tp > &__w)
• template<class _Dom >
  __Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _-
  Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_-
  type >::result_type > operator|| (const __Expr< _Dom, typename _-
  Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-
  _v)
• template<class _Dom >
  __Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _-
  Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_-
  type >::result_type > operator|| (const __Expr< _Dom, typename _-
  Dom::value_type > &__v, const typename _Dom::value_type &__t)
• template<class _Dom >
  __Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_-
  type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type
  >::result_type > operator|| (const typename _Dom::value_type &__t, const _-
  Expr< _Dom, typename _Dom::value_type > &__v)
• template<class _Dom >
  __Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_-
  type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type
  >::result_type > operator|| (const valarray< typename _Dom::value_type >
  &__v, const __Expr< _Dom, typename _Dom::value_type > &__e)
• template<typename _Tp >
  __Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename
  __fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >
  &__v, const _Tp &__t)

```

- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > operator|| (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `_Ios_Iostate operator~ (_Ios_Iostate __a)`
- `_Ios_Openmode operator~ (_Ios_Openmode __a)`
- `_Ios_Fmtflags operator~ (_Ios_Fmtflags __a)`
- `template<typename... _Elements>`
`tuple< typename __pa_add_rvalue_reference< _Elements >::__type...>`
`pack_arguments (_Elements &&...__args)`
- `template<typename _RAIter, typename _Compare >`
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last)`
- `template<typename _Iter, typename _RAIter >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator`
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator`
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result)`
- `template<typename _BIter, typename _Predicate >`
`_BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _-`
`Predicate __pred)`

- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _-`
`Predicate)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, type-`
`name _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __-`
`first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`
`out_false, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _Tp >`
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _RAIter >`
`void pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_-`
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _-`
`Dom >, typename _Dom::value_type > pow (const typename _Dom::value_-`
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_-`
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`
`>::__type, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_-`
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-`
`_v)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > pow (const valarray< typename _`
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `float pow (float __x, float __y)`
- `long double pow (long double __x, long double __y)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _BidirectionalIterator >`
`__gnu_cxx::__enable_if< __is_iterator< _BidirectionalIterator >::__value, _`
`BidirectionalIterator >::__type prev (_BidirectionalIterator __x, typename`
`iterator_traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __`
`last)`
- `template<typename _BIter >`
`bool prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __`
`last, _Compare __comp)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type proj (_Tp __x)`
- `template<typename _Tp >`
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__`
`x)(_Arg1, _Arg2))`

- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RAIter >`
`void push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _MoneyT >`
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __-`
`intl=false)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RAIter, typename _Generator >`
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _RandomNumberGenerator &&__rand)`
- `template<typename _Tp >`
`_Tp real (const complex< _Tp > &__z)`
- `template<typename _FIter, typename _Tp >`
`_FIter remove (_FIter, _FIter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__value)`
- `template<typename _IIter, typename _OIter, typename _Tp >`
`_OIter remove_copy (_IIter, _IIter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`
`_OIter remove_copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`
`_FIter remove_if (_FIter, _FIter, _Predicate)`

- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last,`
`_Predicate __pred)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`
`old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _-`
`Tp >`
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate _-`
`__pred, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`
`void rethrow_if_nested (const _Ex &__ex)`
- `template<typename _Tp >`
`void return_temporary_buffer (_Tp *__p)`
- `template<typename _BidirectionalIterator >`
`void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter >`
`void reverse (_BIter, _BIter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator reverse_copy (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _BIter, typename _OIter >`
`_OIter reverse_copy (_BIter, _BIter, _OIter)`
- `ios_base & right (ios_base &__base)`
- `template<typename _Filter >`
`void rotate (_Filter, _Filter, _Filter)`

- `template<typename _ForwardIterator >`
`void rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __-`
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _Filter, typename _OIter >`
`_OIter rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `ios_base & scientific (ios_base & __base)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __-`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __-`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`
`__predicate)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`
`_Filter search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _-`
`BinaryPredicate >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _-`
`Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _-`
`Integer __count, const _Tp & __val)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`
`_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `new_handler set_new_handler (new_handler) throw ()`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator,`
`typename _Compare >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`
`_OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`
`_OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `terminate_handler set_terminate (terminate_handler) throw ()`
- `unexpected_handler set_unexpected (unexpected_handler) throw ()`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT >`
`_Setfill<_CharT> setfill (_CharT __c)`

- `_Setiosflags` [setiosflags](#) ([ios_base::fmtflags](#) __mask)
- `_Setprecision` [setprecision](#) (int __n)
- `_Setw` [setw](#) (int __n)
- [ios_base](#) & [showbase](#) ([ios_base](#) &__base)
- [ios_base](#) & [showpoint](#) ([ios_base](#) &__base)
- [ios_base](#) & [showpos](#) ([ios_base](#) &__base)
- `template<typename _RAIter, typename _UGenerator >`
void **shuffle** (_RAIter, _RAIter, _UGenerator &)
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
void **shuffle** (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &__g)
- long double **sin** (long double __x)
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type` **sin** (`_Tp` __x)
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp >` **sin** (const [valarray](#)< _Tp > &__v)
- `template<typename _Tp >`
[complex](#)< _Tp > **sin** (const [complex](#)< _Tp > &)
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type >` **sin** (const `_Expr`< _Dom, typename _Dom::value_type > &__e)
- float **sin** (float __x)
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type >` **sinh** (const `_Expr`< _Dom, typename _Dom::value_type > &__e)
- float **sinh** (float __x)
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type` **sinh** (`_Tp` __x)
- long double **sinh** (long double __x)
- `template<typename _Tp >`
[complex](#)< _Tp > **sinh** (const [complex](#)< _Tp > &)
- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp >` **sinh** (const [valarray](#)< _Tp > &__v)
- [ios_base](#) & [skipws](#) ([ios_base](#) &__base)
- `template<typename _RAIter, typename _Compare >`
void **sort** (_RAIter, _RAIter, _Compare)
- `template<typename _RAIter >`
void **sort** (_RAIter, _RAIter)
- `template<typename _RandomAccessIterator, typename _Compare >`
void **sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)

- `template<typename _RandomAccessIterator >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`void sort_heap (_RAIter, _RAIter)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`sqrt (_Tp __x)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp`
`> &__v)`
- `float sqrt (float __x)`
- `template<typename _Tp >`
`complex< _Tp > sqrt (const complex< _Tp > &)`
- `long double sqrt (long double __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Predicate __pred)`
- `template<typename _BIter, typename _Predicate >`
`_BIter stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `double stod (const string &__str, size_t * __idx=0)`
- `float stof (const string &__str, size_t * __idx=0)`
- `int stoi (const string &__str, size_t * __idx=0, int __base=10)`

- long **stol** (const [string](#) &__str, size_t *__idx=0, int __base=10)
- long double **stold** (const [string](#) &__str, size_t *__idx=0)
- long long **stoll** (const [string](#) &__str, size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const [string](#) &__str, size_t *__idx=0, int __base=10)
- unsigned long long **stoull** (const [string](#) &__str, size_t *__idx=0, int __base=10)
- char * **strchr** (char *__s, int __n)
- char * **strpbrk** (char *__s1, const char *__s2)
- char * **strrchr** (char *__s, int __n)
- char * **strstr** (char *__s1, const char *__s2)
- template<typename _Tp, typename _Alloc >
void **swap** ([deque](#)< _Tp, _Alloc > &__x, [deque](#)< _Tp, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code >
void **swap** (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)
- template<typename _Res >
void **swap** ([promise](#)< _Res > &__x, [promise](#)< _Res > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc >
void **swap** ([unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__x, [unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _T1, class _T2 >
void **swap** ([pair](#)< _T1, _T2 > &__x, [pair](#)< _T1, _T2 > &__y)
- template<typename _Tp, size_t _Nm >
void **swap** (_Tp(&)[_Nm], _Tp(&)[_Nm])
- template<typename _Key, typename _Compare, typename _Alloc >
void **swap** ([set](#)< _Key, _Compare, _Alloc > &__x, [set](#)< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq >
void **swap** ([queue](#)< _Tp, _Seq > &__x, [queue](#)< _Tp, _Seq > &__y)
- template<typename _Tp >
void **swap** (_Tp &__a, _Tp &__b)
- template<class _Value, class _Hash, class _Pred, class _Alloc >
void **swap** ([unordered_multiset](#)< _Value, _Hash, _Pred, _Alloc > &__x, [unordered_multiset](#)< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >
void **swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Mutex >
void **swap** ([unique_lock](#)< _Mutex > &__x, [unique_lock](#)< _Mutex > &__y)
- template<typename... _Elements >
void **swap** ([tuple](#)< _Elements... > &__x, [tuple](#)< _Elements... > &__y)

- `template<typename _Tp, typename _Tp_Deleter >`
`void swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _Tp_Deleter > &__y)`
- `template<typename _Bi_iter, typename _Allocator >`
`void swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results< _Bi_iter, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `void swap (thread &__x, thread &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`
- `template<typename _Res, typename... _ArgTypes>`
`void swap (packaged_task< _Res(_ArgTypes...) > &__x, packaged_task< _Res(_ArgTypes...) > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Tp >`
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`
- `template<typename _Res, typename... _Args>`
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Seq >`
`void swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`
`void swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y)`
- `template<typename _Tp >`
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `_GLIBCXX_CONST const error_category & system_category () throw ()`
- `template<typename _Tp >`
`complex< _Tp > tan (const complex< _Tp > &)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `long double tan (long double __x)`
- `float tan (float __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tan (_Tp __x)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`complex< _Tp > tanh (const complex< _Tp > &)`

- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`
`tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float tanh (float __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`tanh (_Tp __x)`
- `long double tanh (long double __x)`
- `void terminate () __attribute__((__noreturn__)) throw ()`
- `const _Rb_tree_node_base *__root throw ()`
- `template<typename _Ex >`
`void throw_with_nested (_Ex __ex)`
- `template<typename... _Elements>`
`tuple< _Elements &...> tie (_Elements &...__args)`
- `string to_string (float __val)`
- `string to_string (unsigned long __val)`
- `string to_string (int __val)`
- `string to_string (long __val)`
- `string to_string (double __val)`
- `string to_string (long double __val)`
- `string to_string (unsigned long long __val)`
- `string to_string (long long __val)`
- `string to_string (unsigned __val)`
- `wstring to_wstring (long double __val)`
- `wstring to_wstring (long long __val)`
- `wstring to_wstring (unsigned long __val)`
- `wstring to_wstring (unsigned long long __val)`
- `wstring to_wstring (double __val)`
- `wstring to_wstring (long __val)`
- `wstring to_wstring (unsigned __val)`
- `wstring to_wstring (int __val)`
- `wstring to_wstring (float __val)`
- `template<typename _CharT >`
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _IIter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_IIter, _IIter, _OIter, _UnaryOperation)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _BinaryOperation >`
`_OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_`
`op)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >
_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >
_OutputIterator transform (_InputIterator __first, _InputIterator __last, _-
OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>
int try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`
- `template<typename... _TElements, typename... _UElements>
tuple< _TElements..., _UElements...> tuple_cat (tuple< _TElements...>
&&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>
tuple< _TElements..., _UElements...> tuple_cat (const tuple< _TElements...>
&&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>
tuple< _TElements..., _UElements...> tuple_cat (tuple< _TElements...>
&&__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>
tuple< _TElements..., _UElements...> tuple_cat (const tuple< _TElements...>
&&__t, const tuple< _UElements...> &__u)`
- `bool uncaught_exception () __attribute__((__pure__)) throw ()`
- `void unexpected () __attribute__((__noreturn__))`
- `template<typename _InputIterator, typename _ForwardIterator >
_ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator _-
last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >
_ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _-
ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >
void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const
_Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >
void uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >
_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _-
BinaryPredicate __binary_pred)`
- `template<typename _FIter >
_FIter unique (_FIter, _FIter)`
- `template<typename _FIter, typename _BinaryPredicate >
_FIter unique (_FIter, _FIter, _BinaryPredicate)`
- `template<typename _ForwardIterator >
_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >
_OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _-
OutputIterator __result)`

- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`
`_OIter unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `ios_base & unitbuf (ios_base &__base)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter upper_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val, _Compare __comp)`
- `ios_base & uppercase (ios_base &__base)`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &__loc)`
- `wchar_t * wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * wcspbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > &__-`
`is)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream<`
`_CharT, _Traits > &__is, bitset< _Nb > &__x)`

- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x, const complex<`
`_Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x, const complex<`
`_Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const complex< _Tp > &__x, const complex<`
`_Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const complex< _Tp > &__x, const complex<`
`_Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__`
`_y)`
- `template<typename _Tp >`
`bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__`
`_y)`

- `template<typename _Tp >`
`bool operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool operator!= (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (_Tp &__t)`
- `template<typename _Tp >`
`reference_wrapper< const _Tp > cref (const _Tp &__t)`
- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >`
`reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
- `template<>`
`basic_istream< char > & operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`

- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`
`bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`

- `template<typename _Ch_type, class _Rx_traits >`
`bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Allocator, class _Rx_traits >`
`bool regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const`

```
basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __-
flags=regex_constants::match_default)
```

Variables

- [enable_if](#)< (!is_member_pointer< _Functor >::value &&!is_function< _Functor >::value &&!is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args...)>::type >::type _-
[_invoke](#) (_Functor &__f, _Args &&...__args)
- static ios_base::Init [__ioinit](#)
- function< void()> [__once_functor](#)
- [std::binder1st](#) [_GLIBCXX_DEPRECATED_ATTR](#)
- const [adopt_lock_t](#) [adopt_lock](#)
- const [defer_lock_t](#) [defer_lock](#)
- const [error_category](#) *const [future_category](#)
- const _Swallow_assign [ignore](#)
- [error_code](#) [make_error_code](#) (errc)
- [error_condition](#) [make_error_condition](#) (errc)
- const nothrow_t [nothrow](#)
- const [try_to_lock_t](#) [try_to_lock](#)

Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the *HOWTO* linked to above.

- [istream](#) [cin](#)
- [ostream](#) [cout](#)
- [ostream](#) [cerr](#)
- [ostream](#) [clog](#)
- [wistream](#) [wcin](#)
- [wostream](#) [wcout](#)
- [wostream](#) [wcerr](#)
- [wostream](#) [wclog](#)

4.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

4.11.2 Typedef Documentation

4.11.2.1 `typedef void(* std::new_handler)()`

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 75 of file `new`.

4.11.2.2 `typedef long long std::streamoff`

Type used by `fpos`, `char_traits<char>`, and `char_traits<wchar_t>`.

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was typedef `long`.

Definition at line 94 of file `postypes.h`.

4.11.2.3 `typedef fpos<mbstate_t> std::streampos`

File position for char streams.

Definition at line 228 of file `postypes.h`.

4.11.2.4 `typedef ptrdiff_t std::streamsize`

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file `postypes.h`.

4.11.2.5 `typedef fpos<mbstate_t> std::u16streampos`

File position for `char16_t` streams.

Definition at line 234 of file `postypes.h`.

4.11.2.6 `typedef fpos<mbstate_t> std::u32streampos`

File position for `char32_t` streams.

Definition at line 236 of file `postypes.h`.

4.11.2.7 `typedef fpos<mbstate_t> std::wstreampos`

File position for `wchar_t` streams.

Definition at line 230 of file postypes.h.

4.11.3 Enumeration Type Documentation

4.11.3.1 anonymous enum

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more. This controls some aspect of the sort routines.

Definition at line 2167 of file stl_algo.h.

4.11.3.2 enum std::float_denorm_style

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the [std::numeric_limits](#) class.

Enumerator:

denorm_indeterminate Indeterminate at compile time whether denormalized values are allowed.

denorm_absent The type does not allow denormalized values.

denorm_present The type allows denormalized values.

Definition at line 170 of file limits.

4.11.3.3 enum std::float_round_style

Describes the rounding style for floating-point types.

This is used in the [std::numeric_limits](#) class.

Enumerator:

round_indeterminate Self-explanatory.

round_toward_zero Self-explanatory.

round_to_nearest To the nearest representable value.

round_toward_infinity Self-explanatory.

round_toward_neg_infinity Self-explanatory.

Definition at line 155 of file limits.

4.11.4 Function Documentation

4.11.4.1 `template<typename _RandomAccessIterator > void
std::__final_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2172 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `sort()`.

4.11.4.2 `template<typename _RandomAccessIterator , typename _Compare >
void std::__final_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2187 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

4.11.4.3 `template<typename _InputIterator , typename _Tp > _InputIterator
std::__find (_InputIterator __first, _InputIterator __last, const _Tp
& __val, input_iterator_tag) [inline]`

This is an overload used by `find()` for the Input Iterator case.

Definition at line 128 of file `stl_algo.h`.

Referenced by `find()`.

4.11.4.4 `template<typename _RandomAccessIterator , typename _Tp >
_RandomAccessIterator std::__find (_RandomAccessIterator
__first, _RandomAccessIterator __last, const _Tp & __val,
random_access_iterator_tag)`

This is an overload used by `find()` for the RAI case.

Definition at line 150 of file `stl_algo.h`.

4.11.4.5 `template<typename _InputIterator , typename _Predicate >
 _InputIterator std::__find_if (_InputIterator __first, _InputIterator
 __last, _Predicate __pred, input_iterator_tag) [inline]`

This is an overload used by `find_if()` for the Input Iterator case.

Definition at line 139 of file `stl_algo.h`.

Referenced by `find_if()`.

4.11.4.6 `template<typename _RandomAccessIterator , typename _Predicate >
 _RandomAccessIterator std::__find_if (_RandomAccessIterator
 __first, _RandomAccessIterator __last, _Predicate __pred,
 random_access_iterator_tag)`

This is an overload used by `find_if()` for the RAI case.

Definition at line 198 of file `stl_algo.h`.

4.11.4.7 `template<typename _InputIterator , typename _Predicate >
 _InputIterator std::__find_if_not (_InputIterator __first,
 _InputIterator __last, _Predicate __pred, input_iterator_tag)
 [inline]`

This is an overload used by `find_if_not()` for the Input Iterator case.

Definition at line 247 of file `stl_algo.h`.

Referenced by `find_if_not()`.

4.11.4.8 `template<typename _RandomAccessIterator , typename _Predicate >
 _RandomAccessIterator std::__find_if_not (_RandomAccessIterator
 __first, _RandomAccessIterator __last, _Predicate __pred,
 random_access_iterator_tag)`

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 258 of file `stl_algo.h`.

4.11.4.9 `template<typename _EuclideanRingElement >
 _EuclideanRingElement std::__gcd (_EuclideanRingElement __m,
 _EuclideanRingElement __n)`

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1487 of file `stl_algo.h`.

4.11.4.10 `template<typename _RandomAccessIterator > void
std::__heap_select (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last)`

This is a helper function for the sort routines.

Definition at line 1895 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

4.11.4.11 `template<typename _RandomAccessIterator , typename _Compare
> void std::__heap_select (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last,
_Compare __comp)`

This is a helper function for the sort routines.

Definition at line 1908 of file `stl_algo.h`.

References `make_heap()`.

4.11.4.12 `template<typename _ForwardIterator , typename
_Predicate , typename _Distance > _ForwardIterator
std::__inplace_stable_partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, _Distance __len)`

This is a helper function...

Definition at line 1772 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

4.11.4.13 `template<typename _RandomAccessIterator > void
std::__inplace_stable_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the stable sorting routines.

Definition at line 3353 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

4.11.4.14 `template<typename _RandomAccessIterator, typename _Compare
> void std::__inplace_stable_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the stable sorting routines.

Definition at line 3372 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insertion_sort()`, and `__merge_without_buffer()`.

4.11.4.15 `template<typename _RandomAccessIterator > void
std::__insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2095 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

4.11.4.16 `template<typename _RandomAccessIterator, typename _Compare
> void std::__insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2118 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

4.11.4.17 `template<typename _RandomAccessIterator, typename _Size
> void std::__introsort_loop (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Size __depth_limit)`

This is a helper function for the sort routine.

Definition at line 2267 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

Referenced by `__introsort_loop()`, and `sort()`.

4.11.4.18 `template<typename _RandomAccessIterator, typename
_Size, typename _Compare > void std::__introsort_loop (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_Size __depth_limit, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2289 of file `stl_algo.h`.

References `__introsort_loop()`, and `__unguarded_partition_pivot()`.

4.11.4.19 `template<typename _Size > _Size std::__lg (_Size __n)
[inline]`

This is a helper function for the sort routines and for [random.tcc](#).

Definition at line 993 of file `stl_algbase.h`.

Referenced by `nth_element()`, `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`, and `sort()`.

4.11.4.20 `template<typename _BidirectionalIterator, typename
_Distance, typename _Pointer > void std::__merge_adaptive (
_BidirectionalIterator __first, _BidirectionalIterator __middle,
_BidirectionalIterator __last, _Distance __len1, _Distance __len2,
_Pointer __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2829 of file `stl_algo.h`.

References `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

4.11.4.21 `template<typename _BidirectionalIterator, typename
_Distance, typename _Pointer, typename _Compare >
void std::__merge_adaptive (_BidirectionalIterator __first,
_BidirectionalIterator __middle, _BidirectionalIterator __last,
_Distance __len1, _Distance __len2, _Pointer __buffer, _Distance
__buffer_size, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2891 of file `stl_algo.h`.

References `__merge_adaptive()`, `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

4.11.4.22 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _BidirectionalIterator3
, typename _Compare > _BidirectionalIterator3
std::__merge_backward (_BidirectionalIterator1 __first1,
_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,
_BidirectionalIterator2 __last2, _BidirectionalIterator3 __result,
_Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2761 of file `stl_algo.h`.

4.11.4.23 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _BidirectionalIterator3
> _BidirectionalIterator3 std::__merge_backward (
_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1,
_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2,
_BidirectionalIterator3 __result)`

This is a helper function for the merge routines.

Definition at line 2726 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

4.11.4.24 `template<typename _BidirectionalIterator , typename _Distance >
void std::__merge_without_buffer (_BidirectionalIterator __first,
_BidirectionalIterator __middle, _BidirectionalIterator __last,
_Distance __len1, _Distance __len2)`

This is a helper function for the merge routines.

Definition at line 2954 of file `stl_algo.h`.

References `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

Referenced by `__inplace_stable_sort()`, `__merge_without_buffer()`, and `inplace_merge()`.

4.11.4.25 `template<typename _BidirectionalIterator , typename _Distance
, typename _Compare > void std::__merge_without_buffer (
 _BidirectionalIterator __first, _BidirectionalIterator __middle,
 _BidirectionalIterator __last, _Distance __len1, _Distance __len2,
 _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2998 of file `stl_algo.h`.

References `__merge_without_buffer()`, `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

4.11.4.26 `template<typename _Iterator , typename _Compare > void
std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator
__c, _Compare __comp)`

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__a`.

Definition at line 100 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.27 `template<typename _Iterator > void std::__move_median_first (
 _Iterator __a, _Iterator __b, _Iterator __c)`

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__a`.

Definition at line 76 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

4.11.4.28 `template<typename _BidirectionalIterator , typename _Predicate
> _BidirectionalIterator std::__partition (_BidirectionalIterator
__first, _BidirectionalIterator __last, _Predicate __pred,
bidirectional_iterator_tag)`

This is a helper function...

Definition at line 1742 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.29 `template<typename _ForwardIterator, typename _Predicate >
 _ForwardIterator std::__partition (_ForwardIterator __first,
 _ForwardIterator __last, _Predicate __pred, forward_iterator_tag
)`

This is a helper function...

Definition at line 1717 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

4.11.4.30 `template<typename _RandomAccessIterator > void std::__reverse (`
`_RandomAccessIterator __first, _RandomAccessIterator __last,`
`random_access_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1407 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.31 `template<typename _BidirectionalIterator > void std::__reverse`
`(_BidirectionalIterator __first, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1387 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__rotate()`, and `reverse()`.

4.11.4.32 `template<typename _ForwardIterator > void std::__rotate`
`(_ForwardIterator __first, _ForwardIterator __middle,`
`_FowardIterator __last, forward_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1501 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`, and `rotate()`.

4.11.4.33 `template<typename _BidirectionalIterator > void std::__rotate (`
`_BidirectionalIterator __first, _BidirectionalIterator __middle,`
`_BidirectionalIterator __last, bidirectional_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1537 of file stl_algo.h.

References `__reverse()`, and `iter_swap()`.

4.11.4.34 `template<typename _RandomAccessIterator > void std::__rotate (`
`_RandomAccessIterator __first, _RandomAccessIterator __middle,`
`_RandomAccessIterator __last, random_access_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1567 of file stl_algo.h.

References `iter_swap()`, `swap()`, and `swap_ranges()`.

4.11.4.35 `template<typename _BidirectionalIterator1 , typename`
`_BidirectionalIterator2 , typename _Distance >`
`_BidirectionalIterator1 std::__rotate_adaptive (`
`_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle,`
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2,`
`_BidirectionalIterator2 __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2797 of file stl_algo.h.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

4.11.4.36 `template<typename _ForwardIterator , typename _Integer`
`, typename _Tp > _ForwardIterator std::__search_n (`
`_FowardIterator __first, _ForwardIterator __last, _Integer`
`__count, const _Tp & __val, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 324 of file stl_algo.h.

Referenced by `search_n()`.

4.11.4.37 `template<typename _ForwardIterator, typename _Integer,
typename _Tp, typename _BinaryPredicate > _ForwardIterator
std::__search_n (_ForwardIterator __first, _ForwardIterator
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for forward iterators.

Definition at line 410 of file `stl_algo.h`.

4.11.4.38 `template<typename _RandomAccessIter, typename _Integer,
typename _Tp, typename _BinaryPredicate > _RandomAccessIter
std::__search_n (_RandomAccessIter __first, _RandomAccessIter
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::random_access_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for random access iterators.

Definition at line 449 of file `stl_algo.h`.

4.11.4.39 `template<typename _RandomAccessIter, typename _Integer,
typename _Tp > _RandomAccessIter std::__search_n (
_RandomAccessIter __first, _RandomAccessIter __last, _Integer
__count, const _Tp & __val, std::random_access_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for random access iterators.

Definition at line 356 of file `stl_algo.h`.

4.11.4.40 `template<typename _ForwardIterator, typename _Pointer,
typename _Predicate, typename _Distance > _ForwardIterator
std::__stable_partition_adaptive (_ForwardIterator __first,
_FowardIterator __last, _Predicate __pred, _Distance __len,
_PPointer __buffer, _Distance __buffer_size)`

This is a helper function...

Definition at line 1797 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

4.11.4.41 `template<typename _RandomAccessIterator > void
std::__unguarded_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last) [inline]`

This is a helper function for the sort routine.

Definition at line 2140 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

4.11.4.42 `template<typename _RandomAccessIterator , typename _Compare
> void std::__unguarded_insertion_sort (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp)
[inline]`

This is a helper function for the sort routine.

Definition at line 2153 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

4.11.4.43 `template<typename _RandomAccessIterator , typename _Compare
> void std::__unguarded_linear_insert (_RandomAccessIterator
__last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2076 of file `stl_algo.h`.

4.11.4.44 `template<typename _RandomAccessIterator > void
std::__unguarded_linear_insert (_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2058 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

4.11.4.45 `template<typename _RandomAccessIterator , typename
_Tp > _RandomAccessIterator std::__unguarded_partition (
_RandomAccessIterator __first, _RandomAccessIterator __last,
const _Tp & __pivot)`

This is a helper function...

Definition at line 2203 of file stl_algo.h.

References iter_swap().

Referenced by __unguarded_partition_pivot().

4.11.4.46 `template<typename _RandomAccessIterator , typename
_Tp , typename _Compare > _RandomAccessIterator
std::__unguarded_partition (_RandomAccessIterator __first,
_RandomAccessIterator __last, const _Tp & __pivot, _Compare
__comp)`

This is a helper function...

Definition at line 2223 of file stl_algo.h.

References iter_swap().

4.11.4.47 `template<typename _RandomAccessIterator >
_RandomAccessIterator std::__unguarded_partition_pivot (
_RandomAccessIterator __first, _RandomAccessIterator __last)
[inline]`

This is a helper function...

Definition at line 2244 of file stl_algo.h.

References __move_median_first(), and __unguarded_partition().

Referenced by __introsort_loop().

4.11.4.48 `template<typename _RandomAccessIterator , typename _Compare
> _RandomAccessIterator std::__unguarded_partition_pivot (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_Compare __comp) [inline]`

This is a helper function...

Definition at line 2256 of file stl_algo.h.

References __move_median_first(), and __unguarded_partition().

4.11.4.49 `template<typename _ForwardIterator, typename _OutputIterator
> _OutputIterator std::__unique_copy (_ForwardIterator
__first, _ForwardIterator __last, _OutputIterator __result,
forward_iterator_tag, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output iterator as result.

Definition at line 1239 of file `stl_algo.h`.

Referenced by `unique_copy()`.

4.11.4.50 `template<typename _InputIterator, typename _OutputIterator
> _OutputIterator std::__unique_copy (_InputIterator __first,
_InputIterator __last, _OutputIterator __result, input_iterator_tag
, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output iterator as result.

Definition at line 1262 of file `stl_algo.h`.

4.11.4.51 `template<typename _InputIterator, typename _ForwardIterator
> _ForwardIterator std::__unique_copy (_InputIterator
__first, _InputIterator __last, _ForwardIterator __result,
input_iterator_tag, forward_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward iterator as result.

Definition at line 1285 of file `stl_algo.h`.

4.11.4.52 `template<typename _ForwardIterator, typename _OutputIterator,
typename _BinaryPredicate > _OutputIterator std::__unique_copy (_ForwardIterator
__first, _ForwardIterator __last, _OutputIterator
__result, _BinaryPredicate __binary_pred, forward_iterator_tag,
output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1306 of file `stl_algo.h`.

4.11.4.53 `template<typename _InputIterator , typename _OutputIterator ,
typename _BinaryPredicate > _OutputIterator std::__unique_copy (
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, _BinaryPredicate __binary_pred, input_iterator_tag ,
output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1335 of file `stl_algo.h`.

4.11.4.54 `template<typename _InputIterator , typename _ForwardIterator ,
typename _BinaryPredicate > _ForwardIterator std::__unique_copy
(_InputIterator __first, _InputIterator __last, _ForwardIterator
__result, _BinaryPredicate __binary_pred, input_iterator_tag ,
forward_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1364 of file `stl_algo.h`.

4.11.4.55 `template<typename _T1 , typename... _Args> void std::_Construct (
_T1 * __p, _Args &&... __args) [inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 73 of file `stl_construct.h`.

4.11.4.56 `template<typename _ForwardIterator > void std::_Destroy (
_ForwardIterator __first, _ForwardIterator __last) [inline]`

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 121 of file `stl_construct.h`.

4.11.4.57 `template<typename _Tp > void std::_Destroy (_Tp * __pointer)
[inline]`

Destroy the object pointed to by a pointer type.

Definition at line 91 of file `stl_construct.h`.

Referenced by `std::deque< _Tp, _Alloc >::_M_fill_initialize()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::reserve()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::~~vector()`.

4.11.4.58 `template<typename _InputIterator , typename _Tp > _Tp
std::accumulate (_InputIterator __first, _InputIterator __last, _Tp
__init) [inline]`

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is *init*. The values are processed in order.

Parameters

first Start of range.

last End of range.

init Starting value to add other values to.

Returns

The final sum.

Definition at line 116 of file `stl_numeric.h`.

4.11.4.59 `template<typename _InputIterator , typename _Tp , typename
_BinaryOperation > _Tp std::accumulate (_InputIterator __first,
_InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
[inline]`

Accumulate values in a range with operation.

Accumulates the values in the range `[first,last)` using the function object *binary_op*. The initial value is *init*. The values are processed in order.

Parameters

first Start of range.

last End of range.

init Starting value to add other values to.

binary_op Function object to accumulate with.

Returns

The final sum.

Definition at line 142 of file `stl_numeric.h`.

4.11.4.60 `template<typename _Tp > _Tp* std::addressof (_Tp & __r)` `[inline]`

declval, from type_traits.

Returns the actual address of the object or function referenced by r, even in the presence of an overloaded operator&.

Parameters

`__r` Reference to an object or function.

Returns

The actual address.

Definition at line 107 of file move.h.

4.11.4.61 `template<typename _InputIterator , typename _OutputIterator >` `_OutputIterator std::adjacent_difference (_InputIterator __first,` `_InputIterator __last, _OutputIterator __result)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using operator-() and writes the result to *result*.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 312 of file stl_numeric.h.

4.11.4.62 `template<typename _InputIterator , typename _OutputIterator` `, typename _BinaryOperation > _OutputIterator` `std::adjacent_difference (_InputIterator __first, _InputIterator` `__last, _OutputIterator __result, _BinaryOperation __binary_op)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using the function object *binary_op* and writes the result to *result*.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 354 of file `stl_numeric.h`.

4.11.4.63 `template<typename _InputIterator , typename _Distance > void
std::advance (_InputIterator & __i, _Distance __n) [inline]`

A generalization of pointer arithmetic.

Parameters

i An input iterator.

n The *delta* by which to change *i*.

Returns

Nothing.

This increments *i* by *n*. For bidirectional and random access iterators, *n* may be negative, in which case *i* is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 169 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `equal_range()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

4.11.4.64 `template<typename _Result , typename _Functor ,
typename... _ArgTypes> _Bind_result<_Result, typename
_Maybe_wrap_member_pointer<_Functor>::type (_ArgTypes...)>
std::bind (_Functor __f, _ArgTypes... __args) [inline]`

Function template for std::bind.

Definition at line 1369 of file functional.

4.11.4.65 `ios_base& std::boolalpha (ios_base & __base) [inline]`

Calls base.setf(ios_base::boolalpha).

Definition at line 794 of file ios_base.h.

References std::ios_base::setf().

Referenced by noboolalpha().

4.11.4.66 `template<typename _Tp > reference_wrapper<const _Tp> std::cref
(const _Tp & __t) [inline]`

Denotes a const reference should be taken to a variable.

Definition at line 435 of file functional.

Referenced by cref().

4.11.4.67 `template<typename _Tp > reference_wrapper<const _Tp> std::cref
(reference_wrapper<_Tp > __t) [inline]`

Partial specialization.

Definition at line 447 of file functional.

References cref().

4.11.4.68 `ios_base& std::dec (ios_base & __base) [inline]`

Calls base.setf(ios_base::dec, ios_base::basefield).

Definition at line 932 of file ios_base.h.

References std::ios_base::setf().

Referenced by operator>>().

4.11.4.69 `template<typename _InputIterator > iterator_traits<_InputIterator>::difference_type std::distance (_InputIterator __first, _InputIterator __last) [inline]`

A generalization of pointer arithmetic.

Parameters

first An input iterator.

last An input iterator.

Returns

The distance between them.

Returns *n* such that *first* + *n* == *last*. This requires that *last* must be reachable from *first*. Note that *n* may be negative.

For random access iterators, this uses their + and – operations and are constant time. For other iterator classes they are linear time.

Definition at line 111 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::M_range_initialize()`, `equal_range()`, `inplace_merge()`, `is_heap_until()`, `std::sub_match<_Bi_iter>::length()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `partition_point()`, `std::match_results<_FwdIterT, _Alloc>::position()`, `__gnu_cxx::random_sample_n()`, and `upper_bound()`.

4.11.4.70 `template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits>& std::endl (basic_ostream<_CharT, _Traits> & __os) [inline]`

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple new-line is desired, leading to poor buffering performance. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this subject.

Definition at line 541 of file `ostream`.

References `flush()`, `std::basic_ostream<_CharT, _Traits>::put()`, and `std::basic_ios<_CharT, _Traits>::widen()`.

4.11.4.71 `template<typename _CharT, typename _Traits >
basic_ostream<_CharT, _Traits>& std::ends (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Write a null character into the output sequence.

Null character is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

Definition at line 552 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits>::put()`.

4.11.4.72 `ios_base& std::fixed (ios_base & __base) [inline]`

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 957 of file `ios_base.h`.

References `std::ios_base::setf()`.

4.11.4.73 `template<typename _CharT, typename _Traits >
basic_ostream<_CharT, _Traits>& std::flush (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 562 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits>::flush()`.

Referenced by `endl()`.

4.11.4.74 `template<typename _Tp > enable_if<!is_lvalue_reference<_
_Tp>::value, _Tp&&>::type std::forward (typename std::identity<
_Tp>::type & __t) [inline]`

`forward` (as per N2835) /// Forward lvalues as rvalues.

Definition at line 65 of file `move.h`.

4.11.4.75 `template<typename _Tp > enable_if<!is_lvalue_reference<_
_Tp>::value, _Tp&&>::type std::forward (typename std::identity<
_Tp>::type && __t) [inline]`

Forward rvalues as rvalues.

Definition at line 71 of file move.h.

```
4.11.4.76 template<typename _MoneyT > _Get_money<_MoneyT>  
std::get_money ( _MoneyT & __mon, bool __intl = false )  
[inline]
```

Extended manipulator for extracting money.

Parameters

mon Either long double or a specialization of `basic_string`.

intl A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *mon*.

Definition at line 256 of file iomanip.

```
4.11.4.77 template<typename _Tp > pair<_Tp*, ptrdiff_t>  
std::get_temporary_buffer ( ptrdiff_t __len )
```

Allocates a temporary buffer.

Parameters

len The number of objects of type *Tp*.

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for *len* adjacent *Tp* objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of sizeof(Tp))*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare *len* with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 84 of file stl_tempbuf.h.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

4.11.4.78 `template<typename _CharT, typename _Traits, typename _Alloc >
 basic_istream< _CharT, _Traits > & std::getline (basic_istream<
 _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc >
 & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

is Input stream.
str Buffer to store into.
delim Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until *delim* is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased. If *delim* was encountered, it is extracted but not stored into *str*.

Definition at line 1068 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.11.4.79 `template<typename _CharT, typename _Traits, typename _Alloc >
 basic_istream< _CharT, _Traits>& std::getline (basic_istream<
 _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc >
 & __str) [inline]`

Read a line from stream into a string.

Parameters

is Input stream.
str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until ' '

' is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*.

Any previous contents of *str* are erased. If end of line was encountered, it is extracted but not stored into *str*.

Definition at line 2675 of file basic_string.h.

References std::basic_ios< _CharT, _Traits >::widen().

4.11.4.80 `template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
basic_istream< _CharT, _Traits > & std::getline (basic_istream<
_CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT,
_Traits, _Alloc, _Base > & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

`__is` Input stream.

`__str` Buffer to store into.

`__delim` Character marking end of line.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

Definition at line 622 of file vstring.tcc.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.11.4.81 `template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
basic_istream< _CharT, _Traits>& std::getline (basic_istream<
_CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT,
_Traits, _Alloc, _Base > & __str) [inline]`

Read a line from stream into a string.

Parameters

`__is` Input stream.

`__str` Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `is` into `__str` until ' '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

Definition at line 2447 of file `vstring.h`.

References `std::basic_ios<_CharT, _Traits>::widen()`.

4.11.4.82 `template<typename _Facet > bool std::has_facet (const locale & __loc) throw ()`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if locale contains a facet of type `Facet`, else false.

Definition at line 91 of file `locale_classes.tcc`.

4.11.4.83 `ios_base& std::hex (ios_base & __base) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 940 of file `ios_base.h`.

References `std::ios_base::setf()`.

4.11.4.84 `template<typename _InputIterator1, typename _InputIterator2,
typename _Tp > _Tp std::inner_product (_InputIterator1 __first1,
_InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
[inline]`

Compute inner product of two ranges.

Starting with an initial value of *init*, multiplies successive elements from the two ranges and adds each product into the accumulated value using operator+(). The values in the ranges are processed in order.

Parameters

first1 Start of range 1.

last1 End of range 1.

first2 Start of range 2.

init Starting value to add other values to.

Returns

The final inner product.

Definition at line 170 of file `stl_numeric.h`.

4.11.4.85 `template<typename _InputIterator1, typename _InputIterator2,
typename _Tp, typename _BinaryOperation1, typename
_BinaryOperation2 > _Tp std::inner_product (_InputIterator1
__first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
__init, _BinaryOperation1 __binary_op1, _BinaryOperation2
__binary_op2) [inline]`

Compute inner product of two ranges.

Starting with an initial value of *init*, applies *binary_op2* to successive elements from the two ranges and accumulates each result into the accumulated value using *binary_op1*. The values in the ranges are processed in order.

Parameters

first1 Start of range 1.

last1 End of range 1.

first2 Start of range 2.

init Starting value to add other values to.

binary_op1 Function object to accumulate with.

binary_op2 Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 202 of file stl_numeric.h.

4.11.4.86 ios_base& std::internal (ios_base & __base) [inline]

Calls base.setf(ios_base::internal, ios_base::adjustfield).

Definition at line 907 of file ios_base.h.

References std::ios_base::setf().

**4.11.4.87 template<typename _ForwardIterator , typename _Tp > void
std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp
__value)**

Create a range of sequentially increasing values.

For each element in the range [first,last) assigns `value` and increments `value` as if by `++value`.

Parameters

first Start of range.

last End of range.

value Starting value.

Returns

Nothing.

Definition at line 81 of file stl_numeric.h.

**4.11.4.88 template<typename _CharT > bool std::isalnum (_CharT __c,
const locale & __loc) [inline]**

Convenience interface to ctype.is(ctype_base::alnum, __c).

**4.11.4.89 template<typename _CharT > bool std::isalpha (_CharT __c,
const locale & __loc) [inline]**

Convenience interface to ctype.is(ctype_base::alpha, __c).

4.11.4.90 `template<typename _CharT > bool std::isctrl (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

4.11.4.91 `template<typename _CharT > bool std::isdigit (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

4.11.4.92 `template<typename _CharT > bool std::isgraph (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

4.11.4.93 `template<typename _CharT > bool std::islower (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

4.11.4.94 `template<typename _CharT > bool std::isprint (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

4.11.4.95 `template<typename _CharT > bool std::ispunct (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

4.11.4.96 `template<typename _CharT > bool std::isspace (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

4.11.4.97 `template<typename _CharT > bool std::isupper (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

4.11.4.98 `template<typename _CharT > bool std::isxdigit (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

4.11.4.99 `ios_base& std::left (ios_base & __base) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 915 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `operator<<()`.

4.11.4.100 `template<class _T1 , class _T2 > pair<typename
__decay_and_strip<_T1>::__type, typename
__decay_and_strip<_T2>::__type> std::make_pair (_T1 && __x,
_T2 && __y) [inline]`

A convenience wrapper for creating a pair from two objects.

Parameters

x The first object.

y The second object.

Returns

A newly-constructed `pair<>` object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 257 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__parallel_merge_advance()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm()`, `__gnu_parallel::__adjacent_find_selector::__M_sequential_algorithm()`, `__gnu_parallel::__find_if_selector::__M_sequential_algorithm()`, `minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::parallel_multiway_merge()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.11.4.101 `ios_base& std::noboolalpha (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::boolalpha)`.

Definition at line 802 of file ios_base.h.

References boolalpha(), and std::ios_base::unsetf().

4.11.4.102 ios_base& std::noshowbase (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showbase).

Definition at line 818 of file ios_base.h.

References showbase(), and std::ios_base::unsetf().

4.11.4.103 ios_base& std::noshowpoint (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showpoint).

Definition at line 834 of file ios_base.h.

References showpoint(), and std::ios_base::unsetf().

4.11.4.104 ios_base& std::noshowpos (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showpos).

Definition at line 850 of file ios_base.h.

References showpos(), and std::ios_base::unsetf().

4.11.4.105 ios_base& std::noskipws (ios_base & __base) [inline]

Calls base.unsetf(ios_base::skipws).

Definition at line 866 of file ios_base.h.

References skipws(), and std::ios_base::unsetf().

4.11.4.106 ios_base& std::nounitbuf (ios_base & __base) [inline]

Calls base.unsetf(ios_base::unitbuf).

Definition at line 898 of file ios_base.h.

References unitbuf(), and std::ios_base::unsetf().

4.11.4.107 ios_base& std::nouppercase (ios_base & __base) [inline]

Calls base.unsetf(ios_base::uppercase).

Definition at line 882 of file ios_base.h.

References `std::ios_base::unsetf()`, and `uppercase()`.

4.11.4.108 `ios_base& std::oct (ios_base & __base) [inline]`

Calls `base.setf(ios_base::oct, ios_base::basefield)`.

Definition at line 948 of file ios_base.h.

References `std::ios_base::setf()`.

4.11.4.109 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test difference of C string and string.

Parameters

lhs C string.

rhs String.

Returns

True if `rhs.compare(lhs) != 0`. False otherwise.

Definition at line 2426 of file basic_string.h.

4.11.4.110 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test difference of string and C string.

Parameters

lhs String.

rhs C string.

Returns

True if `lhs.compare(rhs) != 0`. False otherwise.

Definition at line 2438 of file basic_string.h.

4.11.4.111 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator!= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test difference of two strings.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs.compare(rhs) != 0*. False otherwise.

Definition at line 2414 of file `basic_string.h`.

4.11.4.112 `template<typename _Tp, typename _Alloc > bool std::operator!= (
const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > &
__y) [inline]`

Based on `operator==`.

Definition at line 1941 of file `stl_deque.h`.

4.11.4.113 `template<typename _Res, typename... _Args> bool std::operator!=
(const function< _Res(_Args...)> & __f, nullptr_t) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`false` if the wrapper has no target, `true` otherwise

This function will not throw an exception.

Definition at line 2153 of file `functional`.

4.11.4.114 `template<typename _Res, typename... _Args> bool std::operator!=
(nullptr_t, const function< _Res(_Args...)> & __f) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2159 of file `functional`.

4.11.4.115 `template<typename _Tp, typename _Alloc > bool std::operator!=(
const list<_Tp, _Alloc > & __x, const list<_Tp, _Alloc > & __y)
[inline]`

Based on operator==.

Definition at line 1587 of file stl_list.h.

4.11.4.116 `template<typename _Key, typename _Tp, typename _Compare ,
typename _Alloc > bool std::operator!=(const map<_Key, _Tp,
_Compare, _Alloc > & __x, const map<_Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on operator==.

Definition at line 861 of file stl_map.h.

4.11.4.117 `template<typename _Key, typename _Tp, typename _Compare ,
typename _Alloc > bool std::operator!=(const multimap<_Key,
_Tp, _Compare, _Alloc > & __x, const multimap<_Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on operator==.

Definition at line 792 of file stl_multimap.h.

4.11.4.118 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator!=(const multiset<_Key, _Compare, _Alloc
> & __x, const multiset<_Key, _Compare, _Alloc > & __y)
[inline]`

Returns !(x == y).

Definition at line 687 of file stl_multiset.h.

4.11.4.119 `template<typename _Tp > bool std::operator!=(const
_Fwd_list_iterator<_Tp > & __x, const _Fwd_list_const_iterator<
_Tp > & __y) [inline]`

Forward list iterator inequality comparison.

Definition at line 266 of file forward_list.h.

4.11.4.120 `template<class _T1 , class _T2 > bool std::operator!= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator==` to find the result.

Definition at line 204 of file `stl_pair.h`.

4.11.4.121 `template<typename _Tp , typename _Seq > bool std::operator!= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Based on `operator==`.

Definition at line 294 of file `stl_queue.h`.

4.11.4.122 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator!= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x == y)`.

Definition at line 700 of file `stl_set.h`.

4.11.4.123 `template<typename _Tp , typename _Seq > bool std::operator!= (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y) [inline]`

Based on `operator==`.

Definition at line 259 of file `stl_stack.h`.

4.11.4.124 `template<typename _Tp , typename _Alloc > bool std::operator!= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]`

Based on `operator==`.

Definition at line 1295 of file `stl_vector.h`.

4.11.4.125 `template<typename _Tp , typename _Alloc > bool std::operator!= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on `operator==`.

Definition at line 1260 of file forward_list.h.

4.11.4.126 `template<size_t _Nb> bitset<_Nb> std::operator& (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1365 of file bitset.

4.11.4.127 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs)`

Concatenate two strings.

Parameters

lhs First string.

rhs Last string.

Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 2297 of file basic_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.

4.11.4.128 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs)`

Concatenate C string and string.

Parameters*lhs* First string.*rhs* Last string.**Returns**New string with value of *lhs* followed by *rhs*.

Definition at line 692 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc>::size().

4.11.4.129 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (_CharT
__lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs)`

Concatenate character and string.

Parameters*lhs* First string.*rhs* Last string.**Returns**New string with *lhs* followed by *rhs*.

Definition at line 708 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc>::size().

4.11.4.130 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const
basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT *
__rhs) [inline]`

Concatenate string and C string.

Parameters*lhs* First string.*rhs* Last string.**Returns**New string with *lhs* followed by *rhs*.

Definition at line 2334 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.

4.11.4.131 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const
basic_string<_CharT, _Traits, _Alloc> & __lhs, _CharT __rhs)
[inline]`

Concatenate string and character.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 2350 of file `basic_string.h`.

4.11.4.132 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator< (const basic_string<_CharT, _Traits, _Alloc
> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs
) [inline]`

Test if string precedes string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2451 of file `basic_string.h`.

4.11.4.133 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator< (const _CharT* __lhs, const basic_string<
_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string precedes string.

Parameters*lhs* C string.*rhs* String.**Returns**True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2475 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.134 `template<typename _Key , typename _Tp , typename _Compare ,
 typename _Alloc > bool std::operator< (const map< _Key, _Tp,
 _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
 _Alloc > & __y) [inline]`

Map ordering relation.

Parameters*x* A map.*y* A map of the same type as *x*.**Returns**True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 854 of file stl_map.h.

4.11.4.135 `template<typename _Key , typename _Tp , typename _Compare ,
 typename _Alloc > bool std::operator< (const multimap< _Key,
 _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
 _Compare, _Alloc > & __y) [inline]`

Multimap ordering relation.

Parameters*x* A multimap.*y* A multimap of the same type as *x*.

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 785 of file `stl_multimap.h`.

```
4.11.4.136  template<typename _Key , typename _Compare , typename _Alloc
              > bool std::operator< ( const multiset< _Key, _Compare, _Alloc
              > & __x, const multiset< _Key, _Compare, _Alloc > & __y )
              [inline]
```

Multiset ordering relation.

Parameters

x A multiset.

y A multiset of the same type as x .

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 680 of file `stl_multiset.h`.

```
4.11.4.137  template<typename _Tp , typename _Alloc > bool std::operator< (
              const forward_list< _Tp, _Alloc > & __lx, const forward_list<
              _Tp, _Alloc > & __ly ) [inline]
```

Forward list ordering relation.

Parameters

lx A forward_list.

ly A forward_list of the same type as lx .

Returns

True iff lx is lexicographically less than ly .

This is a total ordering relation. It is linear in the size of the forward lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1252 of file `forward_list.h`.

References `lexicographical_compare()`.

4.11.4.138 `template<class _T1 , class _T2 > bool std::operator< (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

[<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>](http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html)

Definition at line 197 of file `stl_pair.h`.

4.11.4.139 `template<typename _Tp , typename _Seq > bool std::operator< (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Queue ordering relation.

Parameters

x A queue.

y A queue of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 288 of file `stl_queue.h`.

4.11.4.140 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator< (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Set ordering relation.

Parameters

x A set.

y A set of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 693 of file `stl_set.h`.

4.11.4.141 `template<typename _Tp , typename _Seq > bool std::operator< (`
`const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y`
`) [inline]`

Stack ordering relation.

Parameters

x A stack.

y A stack of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 253 of file `stl_stack.h`.

4.11.4.142 `template<typename _Tp , typename _Alloc > bool std::operator< (`
`const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &`
`__y) [inline]`

Vector ordering relation.

Parameters

x A vector.

y A vector of the same type as *x*.

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1288 of file `stl_vector.h`.

References `lexicographical_compare()`.

4.11.4.143 `template<typename _Tp, typename _Alloc > bool std::operator< (`
`const list<_Tp, _Alloc> & __x, const list<_Tp, _Alloc> & __y)`
`[inline]`

List ordering relation.

Parameters

x A list.

y A list of the same type as x .

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1580 of file `stl_list.h`.

References `lexicographical_compare()`.

4.11.4.144 `template<typename _CharT, typename _Traits, typename _Alloc`
`> bool std::operator< (const basic_string<_CharT, _Traits, _Alloc`
`> & __lhs, const _CharT * __rhs) [inline]`

Test if string precedes C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2463 of file `basic_string.h`.

4.11.4.145 `template<typename _Tp, typename _Alloc> bool std::operator< (const deque< _Tp, _Alloc> & __x, const deque< _Tp, _Alloc> & __y) [inline]`

Deque ordering relation.

Parameters

x A deque.

y A deque of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1933 of file `stl_deque.h`.

References `lexicographical_compare()`.

4.11.4.146 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits> & __out, const char * __s) [inline]`

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at `s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 508 of file ostream.

```
4.11.4.147  template<typename _CharT , typename _Traits >
              basic_ostream<_CharT, _Traits>& std::operator<< (
              basic_ostream< _CharT, _Traits > & __out, _CharT __c )
              [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 449 of file ostream.

```
4.11.4.148  template<typename _CharT , typename _Traits , typename
              _Alloc > basic_ostream<_CharT, _Traits>& std::operator<< (
              basic_ostream< _CharT, _Traits > & __os, const basic_string<
              _CharT, _Traits, _Alloc > & __str ) [inline]
```

Write string to a stream.

Parameters

os Output stream.

str String to write out.

Returns

Reference to the output stream.

Output characters of *str* into *os* following the same rules as for writing a C string.

Definition at line 2634 of file `basic_string.h`.

```
4.11.4.149  template<typename _CharT , typename _Traits , typename
              _Tp > basic_ostream<_CharT, _Traits>& std::operator<< (
              basic_ostream< _CharT, _Traits > && __os, const _Tp & __x )
              [inline]
```

Generic inserter for rvalue stream.

Parameters

- os* An input stream.
- x* A reference to the object being inserted.

Returns

os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 579 of file `ostream`.

```
4.11.4.150  template<class _Traits > basic_ostream<char, _Traits>&
              std::operator<< ( basic_ostream< char, _Traits > & __out, char
              __c ) [inline]
```

Character inserters.

Parameters

- out* An output stream.
- c* A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by `[22.2.2.2.2]`). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 460 of file `ostream`.

4.11.4.151 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out, const
unsigned char * __s) [inline]`

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 526 of file ostream.

4.11.4.152 `template<class _CharT , class _Traits , size_t _Nb>
std::basic_ostream<_CharT, _Traits>& std::operator<<(
std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb
> & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1470 of file bitset.

References `std::__ctype_abstract_base< _CharT >::widen()`.

4.11.4.153 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out, signed
char __c) [inline]`

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 466 of file `ostream`.

```
4.11.4.154 template<typename _CharT, typename _Traits >
          basic_ostream<_CharT, _Traits>& std::operator<< (
          basic_ostream<_CharT, _Traits> & __out, char __c )
          [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 454 of file `ostream`.

```
4.11.4.155 template<class _Traits > basic_ostream<char, _Traits>&
          std::operator<< ( basic_ostream<char, _Traits> & __out, const
          signed char * __s ) [inline]
```

String inserters.

Parameters

out An output stream.
s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 521 of file ostream.

```
4.11.4.156 template<typename _CharT , typename _Traits , typename
    _Alloc , template< typename, typename, typename > class
    _Base> basic_ostream<_CharT, _Traits>& std::operator<<
    ( basic_ostream<_CharT, _Traits > & __os, const
    __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base > &
    __str ) [inline]
```

Write string to a stream.

Parameters

__os Output stream.
__str String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into *os* following the same rules as for writing a C string.

Definition at line 2401 of file vstring.h.

```
4.11.4.157 template<typename _CharT , typename _Traits > basic_ostream<
    _CharT, _Traits > & std::operator<< ( basic_ostream<_CharT,
    _Traits > & __out, const char * __s )
```

String inserters.

Parameters

out An output stream.
s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 321 of file ostream.tcc.

References `std::ios_base::badbit`.

```
4.11.4.158  template<typename _CharT , typename _Traits >
             basic_ostream<_CharT, _Traits>& std::operator<< (
             basic_ostream<_CharT, _Traits> & __out, const _CharT* __s )
             [inline]
```

String inserters.

Parameters

out An output stream.
s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 491 of file ostream.

4.11.4.159 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out,
unsigned char __c) [inline]`

Character inserters.

Parameters

out An output stream.
c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 471 of file `ostream`.

4.11.4.160 `template<typename _CharT, typename _Traits, typename
_Alloc > bool std::operator<=(const _CharT * __lhs, const
basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string doesn't follow string.

Parameters

lhs C string.
rhs String.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2549 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.161 `template<typename _Tp, typename _Alloc > bool std::operator<=
(const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y
) [inline]`

Based on `operator<`.

Definition at line 1599 of file stl_list.h.

4.11.4.162 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator<= (const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on operator<.

Definition at line 875 of file stl_map.h.

4.11.4.163 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator<= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 806 of file stl_multimap.h.

4.11.4.164 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator<= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns !(y < x).

Definition at line 701 of file stl_multiset.h.

4.11.4.165 `template<typename _Tp , typename _Alloc > bool std::operator<=
(const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 1281 of file forward_list.h.

4.11.4.166 `template<class _T1 , class _T2 > bool std::operator<= (const
pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses operator< to find the result.

Definition at line 216 of file stl_pair.h.

4.11.4.167 `template<typename _Tp , typename _Seq > bool std::operator<= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Based on operator<.

Definition at line 306 of file stl_queue.h.

4.11.4.168 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator<= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(y < x).

Definition at line 714 of file stl_set.h.

4.11.4.169 `template<typename _Tp , typename _Seq > bool std::operator<= (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y) [inline]`

Based on operator<.

Definition at line 271 of file stl_stack.h.

4.11.4.170 `template<typename _Tp , typename _Alloc > bool std::operator<= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 1307 of file stl_vector.h.

4.11.4.171 `template<typename _Tp , typename _Alloc > bool std::operator<= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 1955 of file stl_deque.h.

4.11.4.172 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test if string doesn't follow C string.

Parameters*lhs* String.*rhs* C string.**Returns**True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2537 of file basic_string.h.

4.11.4.173 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator<= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test if string doesn't follow string.

Parameters*lhs* First string.*rhs* Second string.**Returns**True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2525 of file basic_string.h.

4.11.4.174 `template<typename _Tp, typename _Alloc > bool std::operator== (
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly)`

Forward list equality comparison.

Parameters*lx* A forward_list*ly* A forward_list of the same type as *lx*.**Returns**

True iff the size and elements of the forward lists are equal.

This is an equivalence relation. It is linear in the size of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 361 of file forward_list.tcc.

References `std::forward_list< _Tp, _Alloc >::cbegin()`, and `std::forward_list< _Tp, _Alloc >::cend()`.

4.11.4.175 `template<typename _Tp , typename _Alloc > bool std::operator==(const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]`

List equality comparison.

Parameters

x A list.

y A list of the same type as *x*.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1551 of file stl_list.h.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

4.11.4.176 `template<typename _Res , typename... _Args> bool std::operator==(nullptr_t, const function< _Res(_Args...)> & __f) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2141 of file functional.

4.11.4.177 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator==(const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Multimap equality comparison.

Parameters

- x* A multimap.
- y* A multimap of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 768 of file `stl_multimap.h`.

4.11.4.178 `template<class _T1 , class _T2 > bool std::operator== (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Two pairs of the same type are equal iff their members are equal.

Definition at line 191 of file `stl_pair.h`.

References `std::pair< _T1, _T2 >::first`, and `std::pair< _T1, _T2 >::second`.

4.11.4.179 `template<typename _Tp , typename _Seq > bool std::operator== (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y) [inline]`

Stack equality comparison.

Parameters

- x* A stack.
- y* A stack of the same type as *x*.

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 235 of file `stl_stack.h`.

4.11.4.180 `template<typename _Tp , typename _Seq > bool std::operator==(const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Queue equality comparison.

Parameters

x A queue.

y A queue of the same type as *x*.

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 270 of file `std_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

4.11.4.181 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator==(const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Map equality comparison.

Parameters

x A map.

y A map of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 837 of file `std_map.h`.

4.11.4.182 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator==(const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Multiset equality comparison.

Parameters

- x* A multiset.
- y* A multiset of the same type as *x*.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 663 of file `stl_multiset.h`.

4.11.4.183 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator==(const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test equivalence of two strings.

Parameters

- lhs* First string.
- rhs* Second string.

Returns

True if *lhs.compare(rhs) == 0*. False otherwise.

Definition at line 2368 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.184 `template<typename _StateT > bool std::operator==(const fpos<
_StateT > & __lhs, const fpos< _StateT > & __rhs) [inline]`

Test if equivalent to another position.

Definition at line 216 of file `postypes.h`.

4.11.4.185 `template<typename _Res , typename... _Args> bool
std::operator==(const function< _Res(_Args...)> & __f, nullptr_t
) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 2135 of file functional.

4.11.4.186 `template<typename _Tp , typename _Alloc > bool std::operator==(
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &
__y) [inline]`

Vector equality comparison.

Parameters

x A vector.

y A vector of the same type as *x*.

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1271 of file stl_vector.h.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equal()`, and `std::vector< _Tp, _Alloc >::size()`.

4.11.4.187 `template<typename _Tp , typename _Alloc > bool std::operator==(
const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > &
__y) [inline]`

Deque equality comparison.

Parameters

x A deque.

y A deque of the same type as *x*.

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1915 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `equal()`, and `std::deque< _Tp, _Alloc >::size()`.

4.11.4.188 `template<typename _CharT, typename _Traits, typename
_Alloc> bool std::operator==(const _CharT* __lhs, const
basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test equivalence of C string and string.

Parameters

lhs C string.

rhs String.

Returns

True if *rhs.compare(lhs) == 0*. False otherwise.

Definition at line 2389 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::compare()`.

4.11.4.189 `template<typename _Tp> bool std::operator==(const
_Fwd_list_iterator< _Tp> & __x, const _Fwd_list_const_iterator<
_Tp> & __y) [inline]`

Forward list iterator equality comparison.

Definition at line 257 of file `forward_list.h`.

4.11.4.190 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator==(const set< _Key, _Compare, _Alloc> &
__x, const set< _Key, _Compare, _Alloc> & __y) [inline]`

Set equality comparison.

Parameters

- x* A set.
- y* A set of the same type as *x*.

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 676 of file stl_set.h.

4.11.4.191 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator==(const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test equivalence of string and C string.

Parameters

- lhs* String.
- rhs* C string.

Returns

True if *lhs.compare(rhs) == 0*. False otherwise.

Definition at line 2401 of file basic_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.192 `template<typename _Tp, typename _Alloc > bool std::operator> (
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &
__y) [inline]`

Based on `operator<`.

Definition at line 1301 of file stl_vector.h.

4.11.4.193 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator> (const basic_string< _CharT, _Traits, _Alloc
> & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs
) [inline]`

Test if string follows string.

Parameters

lhs First string.
rhs Second string.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2488 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.194 `template<typename _CharT, typename _Traits, typename _Alloc
 > bool std::operator> (const _CharT * __lhs, const basic_string<
 _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string follows string.

Parameters

lhs C string.
rhs String.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2512 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.195 `template<typename _Tp, typename _Seq > bool std::operator> (
 const stack<_Tp, _Seq> & __x, const stack<_Tp, _Seq> & __y
) [inline]`

Based on `operator<`.

Definition at line 265 of file `stl_stack.h`.

4.11.4.196 `template<typename _Tp, typename _Alloc > bool std::operator> (
 const deque<_Tp, _Alloc> & __x, const deque<_Tp, _Alloc> &
 __y) [inline]`

Based on `operator<`.

Definition at line 1948 of file `stl_deque.h`.

4.11.4.197 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator> (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns $y < x$.

Definition at line 694 of file `stl_multiset.h`.

4.11.4.198 `template<typename _Tp , typename _Seq > bool std::operator> (
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y) [inline]`

Based on `operator<`.

Definition at line 300 of file `stl_queue.h`.

4.11.4.199 `template<typename _Tp , typename _Alloc > bool std::operator> (
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on `operator<`.

Definition at line 1267 of file `forward_list.h`.

4.11.4.200 `template<class _T1 , class _T2 > bool std::operator> (const pair<
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator<` to find the result.

Definition at line 210 of file `stl_pair.h`.

4.11.4.201 `template<typename _Tp , typename _Alloc > bool std::operator> (
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y)
[inline]`

Based on `operator<`.

Definition at line 1593 of file `stl_list.h`.

4.11.4.202 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator> (const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on operator<.

Definition at line 868 of file stl_map.h.

4.11.4.203 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator> (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 799 of file stl_multimap.h.

4.11.4.204 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator> (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns $y < x$.

Definition at line 707 of file stl_set.h.

4.11.4.205 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator> (const basic_string< _CharT, _Traits, _Alloc
> & __lhs, const _CharT * __rhs) [inline]`

Test if string follows C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2500 of file basic_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.206 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator>= (const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on operator<.

Definition at line 882 of file stl_map.h.

4.11.4.207 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator>= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test if string doesn't precede C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2574 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.208 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc >
& __y) [inline]`

Based on operator<.

Definition at line 1313 of file stl_vector.h.

4.11.4.209 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y
) [inline]`

Based on operator<.

Definition at line 1605 of file stl_list.h.

4.11.4.210 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator>= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 813 of file `stl_multimap.h`.

4.11.4.211 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on `operator<`.

Definition at line 1274 of file `forward_list.h`.

4.11.4.212 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc >
& __y) [inline]`

Based on `operator<`.

Definition at line 1962 of file `stl_deque.h`.

4.11.4.213 `template<typename _Tp , typename _Seq > bool std::operator>= (
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]`

Based on `operator<`.

Definition at line 277 of file `stl_stack.h`.

4.11.4.214 `template<class _T1 , class _T2 > bool std::operator>= (const
pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses `operator<` to find the result.

Definition at line 222 of file `stl_pair.h`.

4.11.4.215 `template<typename _Tp , typename _Seq > bool std::operator>= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Based on operator<.

Definition at line 312 of file stl_queue.h.

4.11.4.216 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if string doesn't precede string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2562 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.217 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(x < y).

Definition at line 721 of file stl_set.h.

4.11.4.218 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string doesn't precede string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2586 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.219 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator>= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns `!(x < y)`.

Definition at line 708 of file `stl_multiset.h`.

4.11.4.220 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_istream< _CharT, _Traits > & std::operator>> (
basic_istream< _CharT, _Traits > & __is, basic_string< _CharT,
_Traits, _Alloc > & __str)`

Read stream into a string.

Parameters

is Input stream.

str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased.

Definition at line 996 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::erase()`, `std::ios_base::getloc()`, `std::basic_string<_CharT, _Traits, _Alloc>::max_size()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

4.11.4.221 `template<class _CharT , class _Traits , size_t _Nb>
std::basic_istream<_CharT, _Traits>& std::operator>> (
std::basic_istream< _CharT, _Traits > & __is, bitset<_Nb > &
__x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1402 of file `bitset`.

References `std::basic_string< _CharT, _Traits, _Alloc >::empty()`, `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

4.11.4.222 `template<typename _CharT , typename _Traits , typename
_Alloc , template< typename, typename, typename
> class _Base> basic_istream< _CharT, _Traits > &
std::operator>> (basic_istream< _CharT, _Traits > & __is,
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &
__str)`

Read stream into a string.

Parameters

`__is` Input stream.

`__str` Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 547 of file `vstring.tcc`.

4.11.4.223 `template<> basic_istream<char>& std::operator>> (
basic_istream< char > & __in, char * __s)`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of *
- *char_type* that can store a terminating *eos*.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

4.11.4.224 `template<class _Traits> basic_istream<char, _Traits>&
std::operator>>(basic_istream< char, _Traits> & __in, signed
char * __s) [inline]`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of *
- *char_type* that can store a terminating *eos*.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 756 of file `istream`.

4.11.4.225 `template<class _Traits > basic_istream<char, _Traits>&
std::operator>> (basic_istream< char, _Traits > & __in, signed
char & __c) [inline]`

Character extractors.

Parameters

- in* An input stream.
- c* A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 709 of file `istream`.

4.11.4.226 `template<class _Traits > basic_istream<char, _Traits>&
std::operator>> (basic_istream< char, _Traits > & __in,
unsigned char * __s) [inline]`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of *
- *char_type* that can store a terminating *eos*.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 751 of file `istream`.

4.11.4.227 `template<typename _CharT , typename _Traits , typename
_Tp > basic_istream<_CharT, _Traits>& std::operator>> (
basic_istream< _CharT, _Traits > && __is, _Tp & __x)
[inline]`

Generic extractor for rvalue stream.

Parameters

- is* An input stream.
- x* A reference to the extraction target.

Returns

is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 847 of file istream.

4.11.4.228 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> & __in, _CharT & __c)`

Character extractors.

Parameters

- in* An input stream.
- c* A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 903 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.11.4.229 `template<class _Traits> basic_istream<char, _Traits> & std::operator>> (basic_istream<char, _Traits> & __in, unsigned char & __c) [inline]`

Character extractors.

Parameters

- in* An input stream.

c A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 704 of file istream.

4.11.4.230 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> & __in, _CharT* __s)`

Character string extractors.

Parameters

in An input stream.

s A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is *the number of elements of the largest array of**
- *char_type that can store a terminating eos.*
- `[27.6.1.2.3]/6`

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 935 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::getloc()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

4.11.4.231 `template<size_t _Nb> bitset<_Nb> std::operator^ (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1383 of file `bitset`.

4.11.4.232 `template<size_t _Nb> bitset<_Nb> std::operator| (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1374 of file `bitset`.

4.11.4.233 `template<typename _InputIterator , typename _OutputIterator ,
typename _BinaryOperation > _OutputIterator std::partial_sum (
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, _BinaryOperation __binary_op)`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

Definition at line 273 of file stl_numeric.h.

4.11.4.234 `template<typename _InputIterator , typename _OutputIterator
> _OutputIterator std::partial_sum (_InputIterator __first,
_InputIterator __last, _OutputIterator __result)`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

Definition at line 233 of file stl_numeric.h.

4.11.4.235 `template<typename _MoneyT > _Put_money<_MoneyT>
std::put_money (const _MoneyT & __mon, bool __intl = false)
[inline]`

Extended manipulator for inserting money.

Parameters

mon Either long double or a specialization of `basic_string`.

intl A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *mon*.

Definition at line 292 of file `iomanip`.

4.11.4.236 `template<typename _Tp > reference_wrapper<_Tp> std::ref (
_Tp & __t) [inline]`

Denotes a reference should be taken to a variable.

Definition at line 429 of file `functional`.

Referenced by `ref()`.

4.11.4.237 `template<typename _Tp > reference_wrapper<_Tp> std::ref (
reference_wrapper<_Tp> __t) [inline]`

Partial specialization.

Definition at line 441 of file `functional`.

References `ref()`.

4.11.4.238 `template<typename _InputIterator , typename _OutputIterator
, typename _Tp > _OutputIterator std::replace_copy (
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, const _Tp & __old_value, const _Tp & __new_value)`

Copy a sequence, replacing each element of one value with another value.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

old_value The value to be replaced.

new_value The replacement value.

Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[first,last)` to the output range `[result,result+(last-first))` replacing elements equal to `old_value` with `new_value`.

Definition at line 3748 of file `stl_algo.h`.

4.11.4.239 `_Resetiosflags` `std::resetiosflags (ios_base::fmtflags __mask)` [inline]

Manipulator for `setf`.

Parameters

mask A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0,mask)`.

Definition at line 63 of file `iomanip`.

4.11.4.240 `template<typename _Tp> void std::return_temporary_buffer (` `_Tp * __p)` [inline]

The companion to [get_temporary_buffer\(\)](#).

Parameters

p A buffer previously allocated by `get_temporary_buffer`.

Returns

None.

Frees the memory pointed to by *p*.

Definition at line 111 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.

4.11.4.241 ios_base& std::right (ios_base & __base) [inline]

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 923 of file `ios_base.h`.

References `std::ios_base::setf()`.

4.11.4.242 ios_base& std::scientific (ios_base & __base) [inline]

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 965 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by operator<<().

4.11.4.243 new_handler std::set_new_handler (new_handler) throw ()

Takes a replacement handler as the argument, returns the previous handler.

4.11.4.244 _Setbase std::setbase (int __base) [inline]

Manipulator for `setf`.

Parameters

base A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *base* is any other value.

Definition at line 124 of file `iomanip`.

4.11.4.245 template<typename _CharT > _Setfill<_CharT> std::setfill (_CharT __c) [inline]

Manipulator for `fill`.

Parameters

c The new fill character.

Sent to a stream object, this manipulator calls `fill(c)` for that object.

Definition at line 162 of file `iomanip`.

4.11.4.246 `_Setiosflags` `std::setiosflags (ios_base::fmtflags __mask)` `[inline]`

Manipulator for `setf`.

Parameters

mask A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *mask*.

Definition at line 93 of file `iomanip`.

4.11.4.247 `_Setprecision` `std::setprecision (int __n)` `[inline]`

Manipulator for `precision`.

Parameters

n The new precision.

Sent to a stream object, this manipulator calls `precision(n)` for that object.

Definition at line 192 of file `iomanip`.

4.11.4.248 `_Setw` `std::setw (int __n)` `[inline]`

Manipulator for `width`.

Parameters

n The new width.

Sent to a stream object, this manipulator calls `width(n)` for that object.

Definition at line 222 of file `iomanip`.

4.11.4.249 `ios_base& std::showbase (ios_base & __base)` `[inline]`

Calls `base.setf(ios_base::showbase)`.

Definition at line 810 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowbase()`.

4.11.4.250 ios_base& std::showpoint (ios_base & __base) [inline]

Calls base.setf(ios_base::showpoint).

Definition at line 826 of file ios_base.h.

References std::ios_base::setf().

Referenced by noshowpoint().

4.11.4.251 ios_base& std::showpos (ios_base & __base) [inline]

Calls base.setf(ios_base::showpos).

Definition at line 842 of file ios_base.h.

References std::ios_base::setf().

Referenced by noshowpos().

4.11.4.252 ios_base& std::skipws (ios_base & __base) [inline]

Calls base.setf(ios_base::skipws).

Definition at line 858 of file ios_base.h.

References std::ios_base::setf().

Referenced by noskipws(), and operator>>().

4.11.4.253 template<class _T1 , class _T2 > void std::swap (pair< _T1, _T2 > & __x, pair< _T1, _T2 > & __y) [inline]

See std::pair::swap().

Definition at line 231 of file stl_pair.h.

4.11.4.254 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > & __x, multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

See [std::multimap::swap\(\)](#).

Definition at line 820 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::swap().

4.11.4.255 `template<typename _CharT, typename _Traits, typename _Alloc >
void std::swap (basic_string< _CharT, _Traits, _Alloc > & __lhs,
basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Swap contents of two strings.

Parameters

lhs First string.

rhs Second string.

Exchanges the contents of *lhs* and *rhs* in constant time.

Definition at line 2599 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

4.11.4.256 `template<typename _Res, typename... _Args> void std::swap (
function< _Res(_Args...)> & __x, function< _Res(_Args...)> &
__y) [inline]`

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 2171 of file `functional`.

Referenced by `__gnu_parallel::LoserTree< false, _Tp, _Compare >::__delete_min_insert()`, `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >::__delete_min_insert()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__qsb_divide()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__rotate()`, `std::regex_traits< _Ch_type >::imbue()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::swap()`.

4.11.4.257 `template<typename _Tp, typename _Alloc > void std::swap (list<
_Tp, _Alloc > & __x, list< _Tp, _Alloc > & __y) [inline]`

See [std::list::swap\(\)](#).

Definition at line 1611 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::swap()`.

4.11.4.258 `template<typename _Key, typename _Compare, typename _Alloc
> void std::swap (set< _Key, _Compare, _Alloc > & __x, set<
_Key, _Compare, _Alloc > & __y) [inline]`

See [std::set::swap\(\)](#).

Definition at line 728 of file stl_set.h.

References `std::set<_Key, _Compare, _Alloc >::swap()`.

4.11.4.259 `template<typename _Key , typename _Compare , typename _Alloc
> void std::swap (multiset< _Key, _Compare, _Alloc > & __x,
multiset< _Key, _Compare, _Alloc > & __y) [inline]`

See [std::multiset::swap\(\)](#).

Definition at line 715 of file stl_multiset.h.

References `std::multiset<_Key, _Compare, _Alloc >::swap()`.

4.11.4.260 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > void std::swap (map< _Key, _Tp, _Compare,
_Alloc > & __x, map< _Key, _Tp, _Compare, _Alloc > & __y)
[inline]`

See [std::map::swap\(\)](#).

Definition at line 889 of file stl_map.h.

References `std::map<_Key, _Tp, _Compare, _Alloc >::swap()`.

4.11.4.261 `template<typename _Tp , typename _Alloc > void std::swap (
deque< _Tp, _Alloc > & __x, deque< _Tp, _Alloc > & __y)
[inline]`

See [std::deque::swap\(\)](#).

Definition at line 1969 of file stl_deque.h.

References `std::deque<_Tp, _Alloc >::swap()`.

4.11.4.262 `template<typename _Tp , typename _Alloc > void std::swap (
vector< _Tp, _Alloc > & __x, vector< _Tp, _Alloc > & __y)
[inline]`

See [std::vector::swap\(\)](#).

Definition at line 1319 of file stl_vector.h.

References `std::vector<_Tp, _Alloc >::swap()`.

4.11.4.263 `template<typename _Tp, typename _Alloc > void std::swap (`
`forward_list< _Tp, _Alloc > & __lx, forward_list< _Tp, _Alloc >`
`& __ly) [inline]`

See [std::forward_list::swap\(\)](#).

Definition at line 1288 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::swap()`.

4.11.4.264 `template<typename _CharT > _CharT std::tolower (_CharT __c,`
`const locale & __loc) [inline]`

Convenience interface to `ctype::tolower(__c)`.

Referenced by `std::regex_traits< _Ch_type >::translate_nocase()`.

4.11.4.265 `template<typename _CharT > _CharT std::toupper (_CharT __c,`
`const locale & __loc) [inline]`

Convenience interface to `ctype::toupper(__c)`.

4.11.4.266 `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy (_InputIterator __first,`
`_InputIterator __last, _ForwardIterator __result) [inline]`

Copies the range `[first,last)` into `result`.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

`result + (first - last)`

Like `copy()`, but does not require an initialized output range.

Definition at line 107 of file `stl_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

4.11.4.267 `template<typename _InputIterator , typename _Size , typename
_ForwardIterator > _ForwardIterator std::uninitialized_copy_n (
_InputIterator __first, _Size __n, _ForwardIterator __result)
[inline]`

Copies the range [first,first+n) into result.

Parameters

first An input iterator.

n The number of elements to copy.

result An output iterator.

Returns

result + n

Like [copy_n\(\)](#), but does not require an initialized output range.

Definition at line 629 of file `stl_uninitialized.h`.

References `__iterator_category()`.

4.11.4.268 `template<typename _ForwardIterator , typename _Tp > void
std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __x) [inline]`

Copies the value x into the range [first,last).

Parameters

first An input iterator.

last An input iterator.

x The source value.

Returns

Nothing.

Like `fill()`, but does not require an initialized output range.

Definition at line 164 of file `stl_uninitialized.h`.

4.11.4.269 `template<typename _ForwardIterator , typename _Size , typename
_Tp > void std::uninitialized_fill_n (_ForwardIterator __first,
_Size __n, const _Tp & __x) [inline]`

Copies the value x into the range [first,first+n).

Parameters

- first* An input iterator.
- n* The number of copies to make.
- x* The source value.

Returns

Nothing.

Like `fill_n()`, but does not require an initialized output range.

Definition at line 218 of file `stl_uninitialized.h`.

4.11.4.270 ios_base& std::unitbuf (ios_base & __base) [inline]

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 890 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nounitbuf()`.

4.11.4.271 ios_base& std::uppercase (ios_base & __base) [inline]

Calls `base.setf(ios_base::uppercase)`.

Definition at line 874 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nouppercase()`.

4.11.4.272 template<typename _Facet > const _Facet & std::use_facet (const locale & __loc)

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws [std::bad_cast](#) if the locale doesn't contain a facet of type `Facet`.

Parameters

- Facet* The facet type to access.
- locale* The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

[`std::bad_cast`](#) if locale doesn't contain a facet of type Facet.

Definition at line 105 of file locale_classes.tcc.

Referenced by `std::regex_traits<_Ch_type>::isctype()`, and `std::regex_traits<_Ch_type>::transform()`.

4.11.4.273 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::ws (basic_istream<_CharT, _Traits> & __is)`

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;

std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling `operator>>` on `cin` and your object. Note that the same effect can be achieved by creating a [`std::basic_istream::sentry`](#) inside your definition of `operator>>`.

Definition at line 996 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::getloc()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.11.5 Variable Documentation

4.11.5.1 `enable_if<(is_pointer<_Function>::value && is_function<typename remove_pointer<_Function>::type>::value), typename result_of<_Function(_Args...)>::type>::type std::__invoke [inline]`

Invoke a function object, which may be either a member pointer or a function object. The first parameter will tell which.

Definition at line 229 of file functional.

4.11.5.2 ostream std::cerr

Linked to standard error (unbuffered).

4.11.5.3 istream std::cin

Linked to standard input.

4.11.5.4 ostream std::clog

Linked to standard error (buffered).

4.11.5.5 ostream std::cout

Linked to standard output.

4.11.5.6 wostream std::wcerr

Linked to standard error (unbuffered).

4.11.5.7 wistream std::wcin

Linked to standard input.

4.11.5.8 wostream std::wclog

Linked to standard error (buffered).

4.11.5.9 wostream std::wcout

Linked to standard output.

4.12 std::__debug Namespace Reference

GNU debug code, replaces standard behavior with debug behavior.

Classes

- class `bitset`
Class `std::bitset` with additional safety/checking/debug instrumentation.
- class `deque`
Class `std::deque` with safety/checking/debug instrumentation.
- class `list`
Class `std::list` with safety/checking/debug instrumentation.
- class `map`
Class `std::map` with safety/checking/debug instrumentation.
- class `multimap`
Class `std::multimap` with safety/checking/debug instrumentation.
- class `multiset`
Class `std::multiset` with safety/checking/debug instrumentation.
- class `set`
Class `std::set` with safety/checking/debug instrumentation.
- class `unordered_map`
Class `std::unordered_map` with safety/checking/debug instrumentation.
- class `unordered_multimap`
Class `std::unordered_multimap` with safety/checking/debug instrumentation.
- class `unordered_multiset`
Class `std::unordered_multiset` with safety/checking/debug instrumentation.
- class `unordered_set`
Class `std::unordered_set` with safety/checking/debug instrumentation.
- class `vector`
Class `std::vector` with safety/checking/debug instrumentation.

Functions

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__-`
`_y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`
`Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > &operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _`
`Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _`
`Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`
`Alloc > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const`
`set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`
`> &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs,`
`const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key,`
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, mul-`
`timap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`
`unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp , typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_-`
`set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key,`
`_Compare, _Allocator > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`
`unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare,`
`_Allocator > &__y)`
- `template<typename _Tp , typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp , typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

4.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior. Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

4.13 `std::__detail` Namespace Reference

Implementation details not part of the namespace `std` interface.

Functions

- `template<class _Iterator >`
[std::iterator_traits](#)< _Iterator >::difference_type **__distance_fw** (_Iterator __first, _Iterator __last, [std::input_iterator_tag](#))
- `template<class _Iterator >`
[std::iterator_traits](#)< _Iterator >::difference_type **__distance_fw** (_Iterator __first, _Iterator __last, [std::forward_iterator_tag](#))
- `template<class _Iterator >`
[std::iterator_traits](#)< _Iterator >::difference_type **__distance_fw** (_Iterator __first, _Iterator __last)
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
__transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)
- `template<typename _Value, bool __cache>`
operator!= (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)
- `template<typename _Value, bool __cache>`
operator!= (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)
- `template<typename _Value, bool __cache>`
operator== (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)
- `template<typename _Value, bool __cache>`
operator== (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)

Variables

- `const unsigned long __prime_list []`

4.13.1 Detailed Description

Implementation details not part of the namespace std interface.

4.14 std::__parallel Namespace Reference

GNU parallel code, replaces standard behavior with parallel behavior.

Classes

- `struct _CRandNumber`

Functor wrapper for `std::rand()`.

Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Tp __accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _-`
`IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`
`_Tp __accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _-`
`BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >`
`_Tp __accumulate_switch (_RAIter __begin, _RAIter __end, _Tp __init, _-`
`BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::Parallelism`
`__parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _Tag >`
`_Tp __accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`
`_Tp __accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`
`_Tp __accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_`
`access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_`
`parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`
`_Tag2 >`
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _-`
`Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper,`
`random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism`
`__parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __-`
`end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _-`
`IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _-`
`OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_`
`tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_`
`tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random_`
`access_iterator_tag)`

- `template<typename _FIterator, typename _IteratorTag >`
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _-`
`IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _-`
`BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _-`
`BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _FIter, typename _IterTag >`
`_FIter __adjacent_find_switch (_FIter, _FIter, _IterTag)`
- `template<typename _FIter, typename _BiPredicate, typename _IterTag >`
`_FIter __adjacent_find_switch (_FIter, _FIter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >`
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_`
`access_iterator_tag)`
- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type __count_if_switch (_RAIter __-`
`begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, gnu_`
`parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _-`
`Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _-`
`Predicate, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type __count_switch (_RAIter __begin,`
`_RAIter __end, const _Tp &__value, random_access_iterator_tag, gnu_`
`parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter`
`__end, const _Tp &__value, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const`
`_Tp &, _IterTag)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`
`>`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __-`
`begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _-`
`IteratorTag >`
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator`
`__begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_`
`iterator_tag, _IteratorTag)`

- `template<typename _Iter , typename _FIterator , typename _BinaryPredicate , typename _IteratorTag1 , typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator _begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter , typename _FIter , typename _IterTag1 , typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _IterTag1, _IterTag2)`
- `template<typename _RAIter , typename _FIter , typename _BiPredicate , typename _IterTag >`
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter , typename _FIter , typename _BiPredicate , typename _IterTag1 , typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter , typename _Predicate , typename _IteratorTag >`
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter , typename _Predicate >`
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter , typename _Predicate , typename _IterTag >`
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter , typename _Tp , typename _IteratorTag >`
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter , typename _Tp >`
`_RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _Iter , typename _Tp , typename _IterTag >`
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter , typename _Function >`
`_Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter , typename _Function , typename _IteratorTag >`
`_Function __for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _Iter , typename _Function , typename _IterTag >`
`_Function __for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OutputIterator , typename _Size , typename _Generator , typename _IteratorTag >`
`_OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`

- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen,`
`random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=`
`__gnu_parallel::parallel_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`
`_OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`
`void __generate_switch (_FIterator __begin, _FIterator __end, _Generator __`
`gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`
`void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen,`
`random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=`
`__gnu_parallel::parallel_balanced)`
- `template<typename _FIter, typename _Generator, typename _IterTag >`
`void __generate_switch (_FIter, _FIter, _Generator, _IterTag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1,`
`typename BinaryFunction2 >`
`_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, Bi-`
`naryFunction1, BinaryFunction2, random_access_iterator_tag, random_-`
`access_iterator_tag, __gnu_parallel::__Parallelism=__gnu_parallel::parallel_-`
`unbalanced)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, type-`
`name _BinaryFunction2, typename _Tag1, typename _Tag2 >`
`_Tp __inner_product_switch (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1,`
`_BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1,`
`typename _BinaryFunction2 >`
`_Tp __inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2`
`__first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __`
`binary_op2, random_access_iterator_tag, random_access_iterator_tag, __gnu_-`
`parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, type-`
`name _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Tp __inner_product_switch (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2,`
`_Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2,`
`_IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1,`
`typename _IteratorTag2 >`
`bool __lexicographical_compare_switch (_IIter1 __begin1, _IIter1 __end1,`
`_IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _`
`IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 _`
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_-`
`access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`
`_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _-`
`Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename`
`_IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _-`
`IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Compare, random_access_iterator_tag, random_access_iterator_tag, random_`
`access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare,`
`typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2`
`__begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _-`
`IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 _-`
`begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random_`
`access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`
`_FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _-`
`Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,`
`typename _IteratorTag2 >`
`pair<_Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`

- template<typename _RAIter1, typename _RAIter2, typename _Predicate >
[pair](#)< _RAIter1, _RAIter2 > **__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, [random_access_iterator_tag](#), [random_access_iterator_tag](#))
- template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >
[pair](#)< _Iter1, _Iter2 > **__mismatch_switch** (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
 _OutputIterator **__partial_sum_switch** (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, [random_access_iterator_tag](#), [random_access_iterator_tag](#))
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >
 _OutputIterator **__partial_sum_switch** (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >
 _OIter **__partial_sum_switch** (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _Iter, typename _OIter, typename _BinaryOper >
 _OIter **__partial_sum_switch** (_Iter, _Iter, _OIter, _BinaryOper, [random_access_iterator_tag](#), [random_access_iterator_tag](#))
- template<typename _FIterator, typename _Predicate, typename _IteratorTag >
 _FIterator **__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate >
 _RAIter **__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, [random_tag](#))
- template<typename _FIter, typename _Predicate, typename _IterTag >
 _FIter **__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate, typename _Tp >
 void **__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, [random_access_iterator_tag](#), [gnu_parallel::Parallelism](#) __parallelism_tag=[gnu_parallel::parallel_balanced](#))
- template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >
 void **__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)
- template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag >
 void **__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)
- template<typename _FIter, typename _Tp, typename _IterTag >
 void **__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _FIterator, typename _Tp, typename _IteratorTag >
 void **__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)

- `template<typename _RAIter, typename _Tp >`
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp &_old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`
`_FIter __search_n_switch (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`
`_FIterator __search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2 >`
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _-`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _-`
`IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _-`
`Predicate >`
`_Output_RAlter __set_difference_switch (_RAIter1 __begin1, _RAIter1 _-`
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _-`
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,
random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename`
`_IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename`
`_IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred,`
`_IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _-`
`Predicate >`
`_Output_RAlter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 _-`
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _-`
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,
random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename`
`_IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _-`
`Predicate >`
`_Output_RAlter __set_symmetric_difference_switch (_RAIter1 __begin1, _-`
`RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter`
`__result, _Predicate __pred, random_access_iterator_tag, random_access_`
`iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename`
`_IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_symmetric_difference_switch (_Iter1 __begin1, _Iter1`
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate`
`__pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename`
`_IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2,`
`_OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate,`
`_IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2`
`__begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _`
`IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`
`Predicate >`
`_Output_RAIter __set_union_switch (_RAIter1 __begin1, _RAIter1 __end1,`
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate`
`__pred, random_access_iterator_tag, random_access_iterator_tag, random_`
`access_iterator_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1,`
`typename _IterTag2 >`
`_OIter __transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _`
`IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`
`_RAOIter __transform1_switch (_RAIter, _RAIter, _RAOIter, _`
`UnaryOperation, random_access_iterator_tag, random_access_iterator_tag,`
`__gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _`
`RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag,`
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=_`
`__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2`
`__result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation`
`>`
`_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3,`
`_BiOperation, random_access_iterator_tag, random_access_iterator_tag,`
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__`
`gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, type-`
`name _Tag1, typename _Tag2, typename _Tag3 >`
`_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _`
`Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _`
`BinaryOperation >`
`_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1,`

- `_RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::__parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >
_OutputIterator transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >
_OIter unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >
_RandomAccess_OIter unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >
_OutputIterator unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >
RandomAccessOutputIterator unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >
_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >
_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >
_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >
_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >
_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >
_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _Tp >
_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::__Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _-`
`BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _-`
`BinaryPredicate __pred)`
- `template<typename _FIter >`
`_FIter adjacent_find (_FIter, _FIter)`
- `template<typename _FIter >`
`_FIter adjacent_find (_FIter, _FIter, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _BiPredicate >`
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate)`
- `template<typename _FIter, typename _BiPredicate >`
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_-
tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`
`const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`
`const _Tp &__value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`
`const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`
`_Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`
`_Predicate __pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`
`_Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_-
parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __-`
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIter >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter, _Iter, _Function)`

- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::Parallelism)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _FIter >`
`_FIter min_element (_FIter, _FIter)`
- `template<typename _FIter >`
`_FIter min_element (_FIter, _FIter, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _FIter, typename _Compare >`
`_FIter min_element (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Compare >`
`_FIter min_element (_FIter, _FIter, _Compare, __gnu_parallel::__Parallelism)`

- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _RAIter >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`

- template<typename _RAIter, typename _Compare >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _-
Compare __comp, [__gnu_parallel::sequential_tag](#))
- template<typename _RAIter >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, [__gnu_parallel::sequential_tag](#))
- template<typename _RAIter, typename _Compare >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _-
Compare __comp)
- template<typename _RAIter >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter **partial_sum** (_Iter, _Iter, _OIter, _BinaryOper)
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __-
result, _BinaryOperation __binary_op)
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __-
result, _BinaryOperation __bin_op, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OutputIterator >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __-
result, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OIter >
_OIter **partial_sum** (_Iter, _Iter, _OIter __result)
- template<typename _Iter, typename _OIter >
_OIter **partial_sum** (_Iter, _Iter, _OIter, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter **partial_sum** (_Iter, _Iter, _OIter, _BinaryOper, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OutputIterator >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __-
result)
- template<typename _Filter, typename _Predicate >
_Filter **partition** (_Filter, _Filter, _Predicate)
- template<typename _Filter, typename _Predicate >
_Filter **partition** (_Filter, _Filter, _Predicate, [__gnu_parallel::sequential_tag](#))
- template<typename _FIterator, typename _Predicate >
_FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred,
[__gnu_parallel::sequential_tag](#))
- template<typename _FIterator, typename _Predicate >
_FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)
- template<typename _RAIter >
void **random_shuffle** (_RAIter __begin, _RAIter __end, [__gnu_parallel::sequential_tag](#))

- `template<typename _RAIter >`
`void random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Tp >`
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`
`const _Tp &__new_value)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_-`
`Parallelism)`
- `template<typename _FIterator, typename _Tp >`
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`
`const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`
`const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_-`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`
`_Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_-`
`Parallelism)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`
`_Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`
`_Tp &__new_value)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, __gnu_-`
`parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`

- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`

- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`

```

    _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)
• template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >
  _OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)
• template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >
  _OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::\_\_Parallelism)
• template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >
  _OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)
• template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >
  _OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OIter, typename _UnaryOperation >
  _OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OIter, typename _UnaryOperation >
  _OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)
• template<typename _Iter, typename _OutputIterator, typename _Predicate >
  _OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)
• template<typename _Iter, typename _OutputIterator >
  _OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OIter >
  _OIter unique_copy (_Iter, _Iter, _OIter)
• template<typename _Iter, typename _OIter >
  _OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OutputIterator >
  _OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)
• template<typename _Iter, typename _OIter, typename _Predicate >
  _OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OIter, typename _Predicate >
  _OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)
• template<typename _Iter, typename _OutputIterator, typename _Predicate >
  _OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)

```

4.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

4.15 std::__profile Namespace Reference

GNU profile code, replaces standard behavior with profile behavior.

Classes

- class [bitset](#)
Class [std::bitset](#) wrapper with performance instrumentation.
- class [deque](#)
Class [std::deque](#) wrapper with performance instrumentation.
- class [list](#)
List wrapper with performance instrumentation.
- class [map](#)
Class [std::map](#) wrapper with performance instrumentation.
- class [multimap](#)
Class [std::multimap](#) wrapper with performance instrumentation.
- class [multiset](#)
Class [std::multiset](#) wrapper with performance instrumentation.
- class [set](#)
Class [std::set](#) wrapper with performance instrumentation.
- class [unordered_map](#)
Class [std::unordered_map](#) wrapper with performance instrumentation.
- class [unordered_multimap](#)
Class [std::unordered_multimap](#) wrapper with performance instrumentation.
- class [unordered_multiset](#)
Unordered_multiset wrapper with performance instrumentation.
- class [unordered_set](#)
Unordered_set wrapper with performance instrumentation.

Functions

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<size_t _Nb>`
`bitset<_Nb> operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<typename _Iterator, typename _Sequence>`
`__iterator_tracker<_Iterator, _Sequence> operator+ (typename __iterator_tracker<_Iterator, _Sequence>::difference_type __n, const __iterator_tracker<_Iterator, _Sequence> &__i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>`
`__iterator_tracker<_IteratorL, _Sequence>::difference_type operator- (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence>`
`__iterator_tracker<_Iterator, _Sequence>::difference_type operator- (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _Tp, typename _Alloc>`
`bool operator< (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`
`bool operator< (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`
`bool operator< (const multiset<_Key, _Compare, _Allocator> &__lhs, const multiset<_Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`
`bool operator< (const set<_Key, _Compare, _Allocator> &__lhs, const set<_Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Tp, typename _Alloc>`
`bool operator< (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>`
`bool operator< (const __iterator_tracker<_IteratorL, _Sequence> &__lhs, const __iterator_tracker<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence>`
`bool operator< (const __iterator_tracker<_Iterator, _Sequence> &__lhs, const __iterator_tracker<_Iterator, _Sequence> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`
`bool operator< (const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Tp, typename _Alloc>`
`bool operator< (const list<_Tp, _Alloc> &__lhs, const list<_Tp, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`

- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`

- bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp, typename _Alloc >
bool **operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool **operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool **operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool **operator>** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator >
bool **operator>** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator>** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator>** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator >
bool **operator>** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool **operator>** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)

- `template<typename _Iterator, typename _Sequence >`
`bool operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`
`> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs,`
`const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const`
`set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`

- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`

4.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

4.16 std::chrono Namespace Reference

ISO C++ 0x entities sub namespace for time and date.

Classes

- struct [duration](#)
duration
- struct [duration_values](#)
duration_values
- struct [system_clock](#)
system_clock
- struct [time_point](#)
time_point
- struct [treat_as_floating_point](#)
treat_as_floating_point

Typedefs

- typedef [system_clock](#) **high_resolution_clock**
- typedef [duration](#)< int, [ratio](#)< 3600 > > **hours**
- typedef [duration](#)< int64_t, [micro](#) > **microseconds**
- typedef [duration](#)< int64_t, [milli](#) > **milliseconds**
- typedef [duration](#)< int, [ratio](#)< 60 > > **minutes**
- typedef [system_clock](#) **monotonic_clock**
- typedef [duration](#)< int64_t, [nano](#) > **nanoseconds**
- typedef [duration](#)< int64_t > **seconds**

Functions

- template<typename _ToDuration , typename _Rep , typename _Period >
[enable_if](#)< __is_duration< _ToDuration >::value, _ToDuration >::type
[duration_cast](#) (const [duration](#)< _Rep, _Period > &__d)
- template<typename _Clock , typename _Duration1 , typename _Duration2 >
bool **operator!=** (const [time_point](#)< _Clock, _Duration1 > &__lhs, const
[time_point](#)< _Clock, _Duration2 > &__rhs)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`bool operator!= (const duration< _Rep1, _Period1 > &__lhs, const duration<`
`_Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type operator% (const duration< _Rep1, _Period1 > &__lhs, const dura-`
`tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, typename enable_if<!__-`
`is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > operator%`
`(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, _Rep2 >::type, _Period >`
`operator* (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep2, _Rep1 >::type, _Period >`
`operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const dura-`
`tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >`
`time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`
`_Period2 > >::type > operator+ (const time_point< _Clock, _Duration1 >`
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Duration2 >`
`time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >,`
`_Duration2 > >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs,`
`const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >`
`time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`
`_Period2 > >::type > operator- (const time_point< _Clock, _Duration1 > &__-`
`lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`common_type< _Duration1, _Duration2 >::type operator- (const time_point<`
`_Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__-`
`rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type operator- (const duration< _Rep1, _Period1 > &__lhs, const dura-`
`tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, typename enable_if<!__-`
`is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > operator/`
`(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _-
Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >
bool operator< (const time_point< _Clock, _Duration1 > &__lhs, const time_-
point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration<
_Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration<
_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >
bool operator<= (const time_point< _Clock, _Duration1 > &__lhs, const
time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >
bool operator== (const time_point< _Clock, _Duration1 > &__lhs, const
time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration<
_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >
bool operator> (const time_point< _Clock, _Duration1 > &__lhs, const time_-
point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration<
_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >
bool operator>= (const time_point< _Clock, _Duration1 > &__lhs, const
time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration<
_Rep2, _Period2 > &__rhs)`
- `template<typename _ToDuration, typename _Clock, typename _Duration >
enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _-
ToDuration > >::type time_point_cast (const time_point< _Clock, _Duration
> &__t)`

4.16.1 Detailed Description

ISO C++ 0x entities sub namespace for time and date.

4.16.2 Typedef Documentation

4.16.2.1 typedef duration<int, ratio<3600> > std::chrono::hours

hours

Definition at line 488 of file chrono.

4.16.2.2 typedef duration<int64_t, micro> std::chrono::microseconds

microseconds

Definition at line 476 of file chrono.

4.16.2.3 typedef duration<int64_t, milli> std::chrono::milliseconds

milliseconds

Definition at line 479 of file chrono.

4.16.2.4 typedef duration<int, ratio< 60> > std::chrono::minutes

minutes

Definition at line 485 of file chrono.

4.16.2.5 typedef duration<int64_t, nano> std::chrono::nanoseconds

nanoseconds

Definition at line 473 of file chrono.

4.16.2.6 typedef duration<int64_t > std::chrono::seconds

seconds

Definition at line 482 of file chrono.

4.16.3 Function Documentation

4.16.3.1 `template<typename _ToDuration , typename _Rep , typename
_Period > enable_if<__is_duration<_ToDuration>::value,
_ToDuration>::type std::chrono::duration_cast (const duration<
_Rep, _Period > & __d) [inline]`

`duration_cast`

Definition at line 155 of file `chrono`.

Referenced by `std::this_thread::sleep_for()`, and `time_point_cast()`.

4.16.3.2 `template<typename _ToDuration , typename _Clock , typename
_Duration > enable_if<__is_duration<_ToDuration>::value, time_
point<_Clock, _ToDuration> >::type std::chrono::time_point_cast (
const time_point< _Clock, _Duration > & __t) [inline]`

`time_point_cast`

Definition at line 550 of file `chrono`.

References `duration_cast()`.

4.17 std::decimal Namespace Reference

ISO/IEC TR 24733 Decimal floating-point arithmetic.

Classes

- class [decimal128](#)
3.2.4 Class [decimal128](#).
- class [decimal32](#)
3.2.2 Class [decimal32](#).
- class [decimal64](#)
3.2.3 Class [decimal64](#).

Functions

- double `decimal128_to_double` ([decimal128](#) __d)

- float **decimal128_to_float** ([decimal128](#) __d)
- long double **decimal128_to_long_double** ([decimal128](#) __d)
- long long **decimal128_to_long_long** ([decimal128](#) __d)
- double **decimal32_to_double** ([decimal32](#) __d)
- float **decimal32_to_float** ([decimal32](#) __d)
- long double **decimal32_to_long_double** ([decimal32](#) __d)
- long long **decimal32_to_long_long** ([decimal32](#) __d)
- double **decimal64_to_double** ([decimal64](#) __d)
- float **decimal64_to_float** ([decimal64](#) __d)
- long double **decimal64_to_long_double** ([decimal64](#) __d)
- long long **decimal64_to_long_long** ([decimal64](#) __d)
- double **decimal_to_double** ([decimal32](#) __d)
- double **decimal_to_double** ([decimal64](#) __d)
- double **decimal_to_double** ([decimal128](#) __d)
- float **decimal_to_float** ([decimal32](#) __d)
- float **decimal_to_float** ([decimal64](#) __d)
- float **decimal_to_float** ([decimal128](#) __d)
- long double **decimal_to_long_double** ([decimal32](#) __d)
- long double **decimal_to_long_double** ([decimal64](#) __d)
- long double **decimal_to_long_double** ([decimal128](#) __d)
- long long **decimal_to_long_long** ([decimal32](#) __d)
- long long **decimal_to_long_long** ([decimal64](#) __d)
- long long **decimal_to_long_long** ([decimal128](#) __d)
- static [decimal128](#) **make_decimal128** (long long __coeff, int __exp)
- static [decimal128](#) **make_decimal128** (unsigned long long __coeff, int __exp)
- static [decimal32](#) **make_decimal32** (long long __coeff, int __exp)
- static [decimal32](#) **make_decimal32** (unsigned long long __coeff, int __exp)
- static [decimal64](#) **make_decimal64** (unsigned long long __coeff, int __exp)
- static [decimal64](#) **make_decimal64** (long long __coeff, int __exp)
- bool **operator!=** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, int __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned int __rhs)
- bool **operator!=** ([decimal32](#) __lhs, long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, long long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, [decimal32](#) __rhs)
- bool **operator!=** (unsigned int __lhs, [decimal32](#) __rhs)
- bool **operator!=** (long __lhs, [decimal32](#) __rhs)
- bool **operator!=** (unsigned long __lhs, [decimal32](#) __rhs)
- bool **operator!=** (long long __lhs, [decimal32](#) __rhs)

- bool **operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal64 __lhs, int __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal64 __lhs, long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal64 __lhs, long long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator!=** (long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator!=** (long long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal128 __lhs, int __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal128 __lhs, long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal128 __lhs, long long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator!=** (long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator!=** (long long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator*** (decimal32 __lhs, int __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator*** (int __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **operator*** (long __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **operator*** (long long __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned long long __lhs, decimal32 __rhs)

- **decimal64 operator*** ([decimal32](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, [decimal32](#) __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, int __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, unsigned int __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, long __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, unsigned long __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, long long __rhs)
- **decimal64 operator*** ([decimal64](#) __lhs, unsigned long long __rhs)
- **decimal64 operator*** (int __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** (unsigned int __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** (long __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** (unsigned long __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** (long long __lhs, [decimal64](#) __rhs)
- **decimal64 operator*** (unsigned long long __lhs, [decimal64](#) __rhs)
- **decimal128 operator*** ([decimal32](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** ([decimal64](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, [decimal32](#) __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, [decimal64](#) __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, int __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, unsigned int __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, long __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, unsigned long __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, long long __rhs)
- **decimal128 operator*** ([decimal128](#) __lhs, unsigned long long __rhs)
- **decimal128 operator*** (int __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** (unsigned int __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** (long __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** (unsigned long __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** (long long __lhs, [decimal128](#) __rhs)
- **decimal128 operator*** (unsigned long long __lhs, [decimal128](#) __rhs)
- **decimal32 operator*** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- **decimal64 operator+** (unsigned long long __lhs, [decimal64](#) __rhs)
- **decimal64 operator+** ([decimal64](#) __rhs)
- **decimal128 operator+** ([decimal32](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator+** ([decimal64](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator+** ([decimal128](#) __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, [decimal32](#) __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, [decimal64](#) __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, int __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, unsigned int __rhs)

- **decimal128 operator+** ([decimal128](#) __lhs, long __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, unsigned long __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, long long __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- **decimal128 operator+** ([decimal128](#) __lhs, unsigned long long __rhs)
- **decimal128 operator+** (int __lhs, [decimal128](#) __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, int __rhs)
- **decimal128 operator+** (unsigned int __lhs, [decimal128](#) __rhs)
- **decimal128 operator+** (long __lhs, [decimal128](#) __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, unsigned int __rhs)
- **decimal128 operator+** (unsigned long __lhs, [decimal128](#) __rhs)
- **decimal128 operator+** (long long __lhs, [decimal128](#) __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, long __rhs)
- **decimal128 operator+** (unsigned long long __lhs, [decimal128](#) __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, unsigned long __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, long long __rhs)
- **decimal32 operator+** ([decimal32](#) __lhs, unsigned long long __rhs)
- **decimal32 operator+** (int __lhs, [decimal32](#) __rhs)
- **decimal32 operator+** (unsigned int __lhs, [decimal32](#) __rhs)
- **decimal32 operator+** (long __lhs, [decimal32](#) __rhs)
- **decimal32 operator+** (unsigned long __lhs, [decimal32](#) __rhs)
- **decimal32 operator+** (long long __lhs, [decimal32](#) __rhs)
- **decimal32 operator+** (unsigned long long __lhs, [decimal32](#) __rhs)
- **decimal64 operator+** ([decimal32](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, [decimal32](#) __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, int __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, unsigned int __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, long __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, unsigned long __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, long long __rhs)
- **decimal64 operator+** ([decimal64](#) __lhs, unsigned long long __rhs)
- **decimal64 operator+** (int __lhs, [decimal64](#) __rhs)
- **decimal64 operator+** (unsigned int __lhs, [decimal64](#) __rhs)
- **decimal64 operator+** (long __lhs, [decimal64](#) __rhs)
- **decimal64 operator+** (unsigned long __lhs, [decimal64](#) __rhs)
- **decimal32 operator+** ([decimal32](#) __rhs)
- **decimal64 operator+** (long long __lhs, [decimal64](#) __rhs)
- **decimal32 operator-** ([decimal32](#) __rhs)
- **decimal64 operator-** ([decimal64](#) __rhs)
- **decimal128 operator-** ([decimal128](#) __rhs)
- **decimal32 operator-** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- **decimal32 operator-** ([decimal32](#) __lhs, int __rhs)

- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, long long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [decimal32 operator-](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator-](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, long __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator-](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) (long __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) (long long __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) (unsigned long long __lhs, [decimal128](#) __rhs)

- [decimal32 operator/](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [decimal32 operator/](#) ([decimal32](#) __lhs, int __rhs)
- [decimal32 operator/](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [decimal32 operator/](#) ([decimal32](#) __lhs, long __rhs)
- [decimal32 operator/](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [decimal32 operator/](#) ([decimal32](#) __lhs, long long __rhs)
- [decimal32 operator/](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [decimal32 operator/](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator/](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator/](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator/](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal128 operator/](#) (long long __lhs, [decimal128](#) __rhs)
- [decimal32 operator/](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator/](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator/](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, long __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator/](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator/](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal128 operator/](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator/](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (long __lhs, [decimal128](#) __rhs)

- [decimal128 operator/](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (int __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (unsigned long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- bool [operator<](#) (unsigned int __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, unsigned int __rhs)
- bool [operator<](#) (unsigned int __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, long __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, long long __rhs)
- bool [operator<](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (unsigned long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, unsigned long __rhs)
- bool [operator<](#) (unsigned long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (long long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, unsigned int __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, int __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, long __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, unsigned int __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, int __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, long long __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, int __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- bool [operator<](#) (int __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, unsigned long __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (int __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (long long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (unsigned int __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (long long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, long __rhs)

- bool **operator<** (decimal64 __lhs, unsigned long __rhs)
- bool **operator<** (decimal128 __lhs, decimal64 __rhs)
- bool **operator<** (decimal64 __lhs, long long __rhs)
- bool **operator<** (decimal64 __lhs, decimal64 __rhs)
- bool **operator==** (int __lhs, decimal128 __rhs)
- bool **operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **operator==** (long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long __rhs)
- bool **operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, long long __rhs)
- bool **operator==** (decimal128 __lhs, long __rhs)
- bool **operator==** (decimal32 __lhs, long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **operator==** (decimal64 __lhs, int __rhs)
- bool **operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **operator==** (long __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, long long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **operator==** (long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal64 __rhs)
- bool **operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator==** (decimal32 __lhs, int __rhs)
- bool **operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **operator==** (decimal32 __lhs, long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal32 __rhs)
- bool **operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **operator==** (long __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, long __rhs)
- bool **operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long long __rhs)

- bool **operator==** (int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, int __rhs)
- bool **operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator>** (decimal32 __lhs, decimal64 __rhs)
- bool **operator>** (decimal64 __lhs, decimal32 __rhs)
- bool **operator>** (decimal64 __lhs, decimal128 __rhs)
- bool **operator>** (long __lhs, decimal32 __rhs)
- bool **operator>** (unsigned long __lhs, decimal32 __rhs)
- bool **operator>** (decimal128 __lhs, int __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>** (decimal64 __lhs, unsigned int __rhs)
- bool **operator>** (unsigned int __lhs, decimal32 __rhs)
- bool **operator>** (int __lhs, decimal64 __rhs)
- bool **operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator>** (decimal32 __lhs, int __rhs)
- bool **operator>** (decimal32 __lhs, decimal32 __rhs)
- bool **operator>** (long long __lhs, decimal64 __rhs)
- bool **operator>** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator>** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator>** (decimal64 __lhs, decimal64 __rhs)
- bool **operator>** (long __lhs, decimal128 __rhs)
- bool **operator>** (int __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, unsigned int __rhs)
- bool **operator>** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator>** (long long __lhs, decimal32 __rhs)
- bool **operator>** (decimal128 __lhs, unsigned int __rhs)
- bool **operator>** (unsigned int __lhs, decimal128 __rhs)
- bool **operator>** (decimal128 __lhs, decimal128 __rhs)
- bool **operator>** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator>** (long long __lhs, decimal128 __rhs)
- bool **operator>** (decimal64 __lhs, unsigned long __rhs)
- bool **operator>** (decimal128 __lhs, long __rhs)
- bool **operator>** (decimal64 __lhs, long __rhs)
- bool **operator>** (decimal128 __lhs, decimal32 __rhs)
- bool **operator>** (decimal128 __lhs, unsigned long __rhs)
- bool **operator>** (decimal64 __lhs, int __rhs)
- bool **operator>** (decimal128 __lhs, long long __rhs)
- bool **operator>** (decimal128 __lhs, decimal64 __rhs)

- bool **operator**> (unsigned long __lhs, [decimal128](#) __rhs)
- bool **operator**> ([decimal64](#) __lhs, long long __rhs)
- bool **operator**> (unsigned int __lhs, [decimal64](#) __rhs)
- bool **operator**> (unsigned long __lhs, [decimal64](#) __rhs)
- bool **operator**> (long __lhs, [decimal64](#) __rhs)
- bool **operator**> ([decimal32](#) __lhs, long __rhs)
- bool **operator**> ([decimal64](#) __lhs, unsigned long long __rhs)
- bool **operator**> (int __lhs, [decimal32](#) __rhs)
- bool **operator**>= (unsigned long __lhs, [decimal32](#) __rhs)
- bool **operator**>= ([decimal64](#) __lhs, int __rhs)
- bool **operator**>= ([decimal128](#) __lhs, int __rhs)
- bool **operator**>= ([decimal32](#) __lhs, unsigned long long __rhs)
- bool **operator**>= ([decimal32](#) __lhs, long __rhs)
- bool **operator**>= ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool **operator**>= ([decimal128](#) __lhs, unsigned long long __rhs)
- bool **operator**>= ([decimal64](#) __lhs, unsigned long __rhs)
- bool **operator**>= ([decimal128](#) __lhs, long __rhs)
- bool **operator**>= ([decimal32](#) __lhs, long long __rhs)
- bool **operator**>= ([decimal64](#) __lhs, unsigned int __rhs)
- bool **operator**>= (long long __lhs, [decimal32](#) __rhs)
- bool **operator**>= ([decimal128](#) __lhs, [decimal128](#) __rhs)
- bool **operator**>= ([decimal64](#) __lhs, [decimal64](#) __rhs)
- bool **operator**>= ([decimal32](#) __lhs, int __rhs)
- bool **operator**>= ([decimal64](#) __lhs, long long __rhs)
- bool **operator**>= (unsigned int __lhs, [decimal32](#) __rhs)
- bool **operator**>= (long long __lhs, [decimal128](#) __rhs)
- bool **operator**>= ([decimal128](#) __lhs, unsigned int __rhs)
- bool **operator**>= (unsigned long long __lhs, [decimal128](#) __rhs)
- bool **operator**>= ([decimal64](#) __lhs, unsigned long long __rhs)
- bool **operator**>= ([decimal128](#) __lhs, unsigned long __rhs)
- bool **operator**>= ([decimal64](#) __lhs, [decimal32](#) __rhs)
- bool **operator**>= (int __lhs, [decimal128](#) __rhs)
- bool **operator**>= ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool **operator**>= (unsigned long long __lhs, [decimal32](#) __rhs)
- bool **operator**>= ([decimal32](#) __lhs, unsigned int __rhs)
- bool **operator**>= ([decimal128](#) __lhs, [decimal32](#) __rhs)
- bool **operator**>= (unsigned long __lhs, [decimal64](#) __rhs)
- bool **operator**>= (unsigned int __lhs, [decimal64](#) __rhs)
- bool **operator**>= ([decimal32](#) __lhs, unsigned long __rhs)
- bool **operator**>= (long __lhs, [decimal128](#) __rhs)
- bool **operator**>= (int __lhs, [decimal64](#) __rhs)
- bool **operator**>= ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool **operator**>= ([decimal128](#) __lhs, long long __rhs)

- bool **operator**>= (int __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, decimal128 __rhs)
- bool **operator**>= (long __lhs, decimal64 __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**>= (long long __lhs, decimal64 __rhs)
- bool **operator**>= (unsigned int __lhs, decimal128 __rhs)
- bool **operator**>= (long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, long __rhs)
- bool **operator**>= (unsigned long __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal64 __rhs)

4.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

4.17.2 Function Documentation

4.17.2.1 long long std::decimal::decimal32_to_long_long (decimal32 __d)

Non-conforming extension: Conversion to integral type.

4.18 std::placeholders Namespace Reference

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

4.18.1 Detailed Description

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

4.19 std::regex_constants Namespace Reference

ISO C++-0x entities sub namespace for regex.

5.1 Regular Expression Syntax Options

- enum `__syntax_option` {
`_S_icode, _S_nosubs, _S_optimize, _S_collate,`
`_S_ECMAScript, _S_basic, _S_extended, _S_awk,`
`_S_grep, _S_egrep, _S_syntax_last` }
- typedef unsigned int `syntax_option_type`
- static const `syntax_option_type` `icode`
- static const `syntax_option_type` `nosubs`
- static const `syntax_option_type` `optimize`
- static const `syntax_option_type` `collate`
- static const `syntax_option_type` `ECMAScript`
- static const `syntax_option_type` `basic`
- static const `syntax_option_type` `extended`
- static const `syntax_option_type` `awk`
- static const `syntax_option_type` `grep`
- static const `syntax_option_type` `egrep`

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `__match_flag` {
`_S_not_bol, _S_not_eol, _S_not_bow, _S_not_eow,`
`_S_any, _S_not_null, _S_continuous, _S_prev_avail,`
`_S_sed, _S_no_copy, _S_first_only, _S_match_flag_last` }
- typedef `std::bitset<_S_match_flag_last>` `match_flag_type`
- static const `match_flag_type` `match_default`
- static const `match_flag_type` `match_not_bol`
- static const `match_flag_type` `match_not_eol`
- static const `match_flag_type` `match_not_bow`
- static const `match_flag_type` `match_not_eow`
- static const `match_flag_type` `match_any`
- static const `match_flag_type` `match_not_null`
- static const `match_flag_type` `match_continuous`
- static const `match_flag_type` `match_prev_avail`
- static const `match_flag_type` `format_default`
- static const `match_flag_type` `format_sed`
- static const `match_flag_type` `format_no_copy`
- static const `match_flag_type` `format_first_only`

5.3 Error Types

- enum `error_type` {
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_`
`complexity`,
`_S_error_stack`, `_S_error_last` }
- static const `error_type error_collate` (`_S_error_collate`)
- static const `error_type error_ctype` (`_S_error_ctype`)
- static const `error_type error_escape` (`_S_error_escape`)
- static const `error_type error_backref` (`_S_error_backref`)
- static const `error_type error_brack` (`_S_error_brack`)
- static const `error_type error_paren` (`_S_error_paren`)
- static const `error_type error_brace` (`_S_error_brace`)
- static const `error_type error_badbrace` (`_S_error_badbrace`)
- static const `error_type error_range` (`_S_error_range`)
- static const `error_type error_space` (`_S_error_space`)
- static const `error_type error_badrepeat` (`_S_error_badrepeat`)
- static const `error_type error_complexity` (`_S_error_complexity`)
- static const `error_type error_stack` (`_S_error_stack`)

4.19.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

4.19.2 Typedef Documentation

4.19.2.1 `typedef std::bitset<_S_match_flag_last> std::regex_constants::match_flag_type`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 190 of file `regex_constants.h`.

4.19.2.2 `typedef unsigned int std::regex_constants::syntax_option_type`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 72 of file `regex_constants.h`.

4.19.3 Enumeration Type Documentation

4.19.3.1 `enum std::regex_constants::__match_flag`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 167 of file `regex_constants.h`.

4.19.3.2 `enum std::regex_constants::__syntax_option`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 46 of file `regex_constants.h`.

4.19.3.3 `enum std::regex_constants::error_type`

The expression contained an invalid collating element name.

Definition at line 43 of file `regex_error.h`.

4.19.4 Function Documentation

4.19.4.1 `static const error_type std::regex_constants::error_backref (_S_error_backref) [static]`

The expression contained an invalid back reference.

**4.19.4.2 static const error_type std::regex_constants::error_badbrace (
_S_error_badbrace) [static]**

The expression contained an invalid range in a { } expression.

**4.19.4.3 static const error_type std::regex_constants::error_badrepeat (
_S_error_badrepeat) [static]**

One of *?+{*was not preceded by a valid regular expression.*

**4.19.4.4 static const error_type std::regex_constants::error_brace (
_S_error_brace) [static]**

The expression contained mismatched { and }

**4.19.4.5 static const error_type std::regex_constants::error_brack (
_S_error_brack) [static]**

The expression contained mismatched [and].

**4.19.4.6 static const error_type std::regex_constants::error_collate (
_S_error_collate) [static]**

The expression contained an invalid collating element name.

**4.19.4.7 static const error_type std::regex_constants::error_complexity (
_S_error_complexity) [static]**

The complexity of an attempted match against a regular expression exceeded a pre-set level.

**4.19.4.8 static const error_type std::regex_constants::error_ctype (
_S_error_ctype) [static]**

The expression contained an invalid character class name.

**4.19.4.9 static const error_type std::regex_constants::error_escape (
_S_error_escape) [static]**

The expression contained an invalid escaped character, or a trailing escape.

4.19.4.10 `static const error_type std::regex_constants::error_paren (` `_S_error_paren) [static]`

The expression contained mismatched (and).

4.19.4.11 `static const error_type std::regex_constants::error_range (` `_S_error_range) [static]`

The expression contained an invalid character range, such as [b-a] in most encodings.

4.19.4.12 `static const error_type std::regex_constants::error_space (` `_S_error_space) [static]`

There was insufficient memory to convert the expression into a finite state machine.

4.19.4.13 `static const error_type std::regex_constants::error_stack (` `_S_error_stack) [static]`

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

4.19.5 Variable Documentation

4.19.5.1 `const syntax_option_type std::regex_constants::awk [static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility awk in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\'`, `\`, and `\ddd` (where ddd is one, two, or three octal digits).

Definition at line 136 of file `regex_constants.h`.

4.19.5.2 `const syntax_option_type std::regex_constants::basic [static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology -- Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 118 of file `regex_constants.h`.

4.19.5.3 const syntax_option_type std::regex_constants::collate [static]

Specifies that character ranges of the form [a-b] should be locale sensitive.

Definition at line 99 of file regex_constants.h.

4.19.5.4 const syntax_option_type std::regex_constants::ECMAScript [static]

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 109 of file regex_constants.h.

4.19.5.5 const syntax_option_type std::regex_constants::egrep [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep when given the -E option in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that newlines are treated as whitespace.

Definition at line 152 of file regex_constants.h.

4.19.5.6 const syntax_option_type std::regex_constants::extended [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 126 of file regex_constants.h.

4.19.5.7 const match_flag_type std::regex_constants::format_default [static]

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript replace function in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 String.prototype.replace. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in [1,9] and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on [01, 99]. If `nn <= match_results::size()` and the *nn*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Definition at line 272 of file `regex_constants.h`.

4.19.5.8 `const match_flag_type std::regex_constants::format_first_only` `[static]`

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 293 of file `regex_constants.h`.

4.19.5.9 `const match_flag_type std::regex_constants::format_no_copy` `[static]`

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 287 of file `regex_constants.h`.

4.19.5.10 `const match_flag_type std::regex_constants::format_sed` `[static]`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology -- Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 280 of file `regex_constants.h`.

4.19.5.11 const syntax_option_type std::regex_constants::grep [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

Definition at line 144 of file `regex_constants.h`.

4.19.5.12 const syntax_option_type std::regex_constants::icase [static]

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 78 of file `regex_constants.h`.

4.19.5.13 const match_flag_type std::regex_constants::match_any [static]

If more than one match is possible then any match is an acceptable result.

Definition at line 227 of file `regex_constants.h`.

4.19.5.14 const match_flag_type std::regex_constants::match_continuous [static]

The expression only matches a sub-sequence that begins at first .

Definition at line 237 of file `regex_constants.h`.

4.19.5.15 const match_flag_type std::regex_constants::match_default [static]

The default matching rules.

Definition at line 195 of file `regex_constants.h`.

4.19.5.16 const match_flag_type std::regex_constants::match_not_bol [static]

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 202 of file `regex_constants.h`.

4.19.5.17 `const match_flag_type std::regex_constants::match_not_bow` `[static]`

The expression `\b` is not matched against the sub-sequence `[first,first)`.

Definition at line 215 of file `regex_constants.h`.

4.19.5.18 `const match_flag_type std::regex_constants::match_not_eol` `[static]`

The last character in the sequence `[first, last)` is treated as though it is not at the end of a line, so the character `(\n)` in the regular expression shall not match `[last, last)`.

Definition at line 209 of file `regex_constants.h`.

4.19.5.19 `const match_flag_type std::regex_constants::match_not_eow` `[static]`

The expression `\b` should not be matched against the sub-sequence `[last,last)`.

Definition at line 221 of file `regex_constants.h`.

4.19.5.20 `const match_flag_type std::regex_constants::match_not_null` `[static]`

The expression does not match an empty sequence.

Definition at line 232 of file `regex_constants.h`.

4.19.5.21 `const match_flag_type std::regex_constants::match_prev_avail` `[static]`

`--first` is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the regular expression algorithms 7.11 and iterators 7.12.

Definition at line 244 of file `regex_constants.h`.

4.19.5.22 `const syntax_option_type std::regex_constants::nosubs` `[static]`

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied [match_results](#) structure.

Definition at line 85 of file `regex_constants.h`.

4.19.5.23 const syntax_option_type std::regex_constants::optimize [static]

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 93 of file regex_constants.h.

4.20 std::rel_ops Namespace Reference

The generated relational operators are sequestered here.

Functions

- template<class _Tp >
bool **operator!=** (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
bool **operator<=** (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
bool **operator>** (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
bool **operator>=** (const _Tp &__x, const _Tp &__y)

4.20.1 Detailed Description

The generated relational operators are sequestered here.

4.20.2 Function Documentation

4.20.2.1 template<class _Tp > bool std::rel_ops::operator!= (const _Tp & __x, const _Tp & __y) [inline]

Defines != for arbitrary types, in terms of ==.

Parameters

- x** A thing.
- y** Another thing.

Returns

- x != y**

This function uses `==` to determine its result.

Definition at line 85 of file `stl_relops.h`.

4.20.2.2 `template<class _Tp> bool std::rel_ops::operator<= (const _Tp & __x, const _Tp & __y) [inline]`

Defines `<=` for arbitrary types, in terms of `<`.

Parameters

x A thing.

y Another thing.

Returns

`x <= y`

This function uses `<` to determine its result.

Definition at line 111 of file `stl_relops.h`.

4.20.2.3 `template<class _Tp> bool std::rel_ops::operator> (const _Tp & __x, const _Tp & __y) [inline]`

Defines `>` for arbitrary types, in terms of `<`.

Parameters

x A thing.

y Another thing.

Returns

`x > y`

This function uses `<` to determine its result.

Definition at line 98 of file `stl_relops.h`.

4.20.2.4 `template<class _Tp> bool std::rel_ops::operator>= (const _Tp & __x, const _Tp & __y) [inline]`

Defines `>=` for arbitrary types, in terms of `<`.

Parameters

- x* A thing.
- y* Another thing.

Returns

x \geq *y*

This function uses $<$ to determine its result.

Definition at line 124 of file `std_relops.h`.

4.21 `std::this_thread` Namespace Reference

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this_thread](#).

Functions

- [thread::id](#) `get_id()`
- `template<typename _Rep, typename _Period >`
`void sleep_for (const chrono::duration< _Rep, _Period > &__rtime)`
- `template<typename _Clock, typename _Duration >`
`void sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)`
- `void yield()`

4.21.1 Detailed Description

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this_thread](#).

4.21.2 Function Documentation

4.21.2.1 `thread::id` `std::this_thread::get_id()` [`inline`]

`get_id`

Definition at line 251 of file `thread`.

4.21.2.2 `template<typename _Rep, typename _Period > void` `std::this_thread::sleep_for (const chrono::duration< _Rep, _Period` `> & __rtime)` [`inline`]

`sleep_for`

Definition at line 270 of file thread.

References `std::chrono::duration_cast()`.

Referenced by `sleep_until()`.

4.21.2.3 `template<typename _Clock , typename _Duration > void
std::this_thread::sleep_until (const chrono::time_point< _Clock,
_Duration > & __atime) [inline]`

`sleep_until`

Definition at line 264 of file thread.

References `sleep_for()`.

4.21.2.4 `void std::this_thread::yield () [inline]`

`yield`

Definition at line 256 of file thread.

4.22 std::tr1 Namespace Reference

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

Namespaces

- namespace [__detail](#)

Classes

- struct [__is_member_pointer_helper](#)
is_member_pointer
- struct [add_const](#)
add_const
- struct [add_cv](#)
add_cv
- struct [add_pointer](#)
add_pointer

- struct [add_volatile](#)
add_volatile
- struct [alignment_of](#)
alignment_of
- struct [array](#)
A standard container for storing a fixed size sequence of elements.
- class [bad_weak_ptr](#)
Exception possibly thrown by [shared_ptr](#).
- struct [extent](#)
extent
- struct [has_virtual_destructor](#)
has_virtual_destructor
- struct [integral_constant](#)
integral_constant
- struct [is_abstract](#)
is_abstract
- struct [is_arithmetic](#)
is_arithmetic
- struct [is_array](#)
is_array
- struct [is_class](#)
is_class
- struct [is_compound](#)
is_compound
- struct [is_const](#)
is_const
- struct [is_empty](#)
is_empty

- struct [is_enum](#)
is_enum
- struct [is_floating_point](#)
is_floating_point
- struct [is_function](#)
is_function
- struct [is_fundamental](#)
is_fundamental
- struct [is_integral](#)
is_integral
- struct [is_member_function_pointer](#)
is_member_function_pointer
- struct [is_member_object_pointer](#)
is_member_object_pointer
- struct [is_object](#)
is_object
- struct [is_pointer](#)
is_pointer
- struct [is_polymorphic](#)
is_polymorphic
- struct [is_same](#)
is_same
- struct [is_scalar](#)
is_scalar
- struct [is_union](#)
is_union
- struct [is_void](#)
is_void

- struct [is_volatile](#)
is_volatile
- struct [rank](#)
rank
- struct [remove_all_extents](#)
remove_all_extents
- struct [remove_const](#)
remove_const
- struct [remove_cv](#)
remove_cv
- struct [remove_extent](#)
remove_extent
- struct [remove_pointer](#)
remove_pointer
- struct [remove_volatile](#)
remove_volatile

Typedefs

- typedef [integral_constant](#)< bool, false > [false_type](#)
- typedef [integral_constant](#)< bool, true > [true_type](#)

Functions

- template<typename _Tp >
[std::complex](#)< _Tp > [__complex_acos](#) (const [std::complex](#)< _Tp > &__z)
- template<typename _Tp >
[std::complex](#)< _Tp > [__complex_acosh](#) (const [std::complex](#)< _Tp > &__z)
- template<typename _Tp >
[std::complex](#)< _Tp > [__complex_asin](#) (const [std::complex](#)< _Tp > &__z)
- template<typename _Tp >
[std::complex](#)< _Tp > [__complex_asinh](#) (const [std::complex](#)< _Tp > &__z)

- `template<typename _Tp >`
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `void __throw_bad_weak_ptr ()`
- `template<typename _Tp >`
`std::complex< _Tp > acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `template<typename _Tp >`
`std::complex< _Tp > asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tp >`
`std::complex< _Tp > atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tpx, typename _Tpy >`
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta (_Tpx __x, _Tpy __y)`
- `float betaf (float __x, float __y)`
- `long double betal (long double __x, long double __y)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)`
- `float comp_ellint_1f (float __k)`
- `long double comp_ellint_1l (long double __k)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)`
- `float comp_ellint_2f (float __k)`
- `long double comp_ellint_2l (long double __k)`

- `template<typename _Tp, typename _Tpn >`
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float comp_ellint_3f (float __k, float __nu)`
- `long double comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hypergf (float __a, float __c, float __x)`
- `long double conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tp >`
`std::complex< _Tp > conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > conj (_Tp __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)`
- `float ellint_1f (float __k, float __phi)`
- `long double ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)`
- `float ellint_2f (float __k, float __phi)`

- long double **ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp >
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type **ellint_3** (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **ellint_3f** (float __k, float __nu, float __phi)
- long double **ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **expint** (_Tp __x)
- float **expintf** (float __x)
- long double **expintl** (long double __x)
- template<typename _Tp >
_Tp **fabs** (const std::complex< _Tp > &__z)
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
_Tp & **get** (array< _Tp, _Nm > &__arr)
- template<std::size_t _Int, class _Tp1, class _Tp2 >
const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & **get** (const std::pair< _Tp1, _Tp2 > &__in)
- template<std::size_t _Int, class _Tp1, class _Tp2 >
tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & **get** (std::pair< _Tp1, _Tp2 > &__in)
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
const _Tp & **get** (const array< _Tp, _Nm > &__arr)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **hermite** (unsigned int __n, _Tp __x)
- float **hermitef** (unsigned int __n, float __x)
- long double **hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type **hyperg** (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **hypergf** (float __a, float __b, float __c, float __x)
- long double **hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **imag** (_Tp)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **laguerre** (unsigned int __n, _Tp __x)
- float **laguerref** (unsigned int __n, float __x)
- long double **laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **legendre** (unsigned int __n, _Tp __x)
- float **legendref** (unsigned int __n, float __x)
- long double **legendrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **norm** (_Tp __x)

- `template<typename _Tp, std::size_t _Nm>`
`bool operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `long double pow (long double __x, long double __y)`
- `double pow (double __x, double __y)`
- `template<typename _Tp >`
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `float pow (float __x, float __y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)`
- `float riemann_zetaf (float __x)`
- `long double riemann_zetal (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`
- `template<typename _Tp, std::size_t _Nm>`
`void swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`

4.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

4.23 std::tr1::__detail Namespace Reference

Implementation details not part of the namespace [std::tr1](#) interface.

4.23.1 Detailed Description

Implementation details not part of the namespace [std::tr1](#) interface.

Chapter 5

Class Documentation

5.1 `__atomic0::atomic_address` Struct Reference

29.4.2, address types

Public Member Functions

- **atomic_address** (void *__v)
- **atomic_address** (const [atomic_address](#) &)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- void * **exchange** (void *__v, memory_order __m=memory_order_seq_cst)
- void * **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)
- void * **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- void * **load** (memory_order __m=memory_order_seq_cst) const
- **operator void *** () const
- void * **operator+=** (ptrdiff_t __d)
- void * **operator-=** (ptrdiff_t __d)
- [atomic_address](#) & **operator=** (const [atomic_address](#) &) volatile
- void * **operator=** (void *__v)
- void **store** (void *__v, memory_order __m=memory_order_seq_cst)

5.1.1 Detailed Description

29.4.2, address types

Definition at line 103 of file atomic_0.h.

The documentation for this struct was generated from the following file:

- [atomic_0.h](#)

5.2 __atomic0::atomic_bool Struct Reference

[atomic_bool](#)

Public Member Functions

- **atomic_bool** (bool __i)
- **atomic_bool** (const [atomic_bool](#) &)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- bool **load** (memory_order __m=memory_order_seq_cst) const
- **operator bool** () const
- [atomic_bool](#) & **operator=** (const [atomic_bool](#) &) volatile
- bool **operator=** (bool __i)
- void **store** (bool __i, memory_order __m=memory_order_seq_cst)

5.2.1 Detailed Description

[atomic_bool](#)

Definition at line 391 of file atomic_0.h.

The documentation for this struct was generated from the following file:

- [atomic_0.h](#)

5.3 `__atomic0::atomic_flag` Struct Reference

[atomic_flag](#)

Inherits `__atomic_flag_base`.

Public Member Functions

- **atomic_flag** (bool __i)
- **atomic_flag** (const [atomic_flag](#) &)
- void **clear** (memory_order __m=memory_order_seq_cst)
- [atomic_flag](#) & **operator=** (const [atomic_flag](#) &) volatile
- bool **test_and_set** (memory_order __m=memory_order_seq_cst)

5.3.1 Detailed Description

[atomic_flag](#)

Definition at line 85 of file `atomic_0.h`.

The documentation for this struct was generated from the following file:

- [atomic_0.h](#)

5.4 `__atomic2::atomic_address` Struct Reference

29.4.2, address types

Public Member Functions

- **atomic_address** (void *__v)
- **atomic_address** (const [atomic_address](#) &)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- void * **exchange** (void *__v, memory_order __m=memory_order_seq_cst)
- void * **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)

- void * **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- void * **load** (memory_order __m=memory_order_seq_cst) const
- **operator void *** () const
- void * **operator+=** (ptrdiff_t __d)
- void * **operator-=** (ptrdiff_t __d)
- [atomic_address](#) & **operator=** (const [atomic_address](#) &) volatile
- void * **operator=** (void *__v)
- void **store** (void *__v, memory_order __m=memory_order_seq_cst)

5.4.1 Detailed Description

29.4.2, address types

Definition at line 81 of file [atomic_2.h](#).

The documentation for this struct was generated from the following file:

- [atomic_2.h](#)

5.5 [__atomic2::atomic_bool](#) Struct Reference

[atomic_bool](#)

Public Member Functions

- **atomic_bool** (bool __i)
- **atomic_bool** (const [atomic_bool](#) &)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- bool **load** (memory_order __m=memory_order_seq_cst) const
- **operator bool** () const
- [atomic_bool](#) & **operator=** (const [atomic_bool](#) &) volatile
- bool **operator=** (bool __i)
- void **store** (bool __i, memory_order __m=memory_order_seq_cst)

5.5.1 Detailed Description

[atomic_bool](#)

Definition at line 391 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic_2.h](#)

5.6 `__atomic2::atomic_flag` Struct Reference

[atomic_flag](#)

Inherits `__atomic_flag_base`.

Public Member Functions

- `atomic_flag` (`bool __i`)
- `atomic_flag` (`const atomic_flag &`)
- `atomic_flag & operator=` (`const atomic_flag &`) volatile

5.6.1 Detailed Description

[atomic_flag](#)

Definition at line 47 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic_2.h](#)

5.7 `__cxxabiv1::__forced_unwind` Class Reference

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

5.7.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

Definition at line 47 of file cxxabi-forced.h.

The documentation for this class was generated from the following file:

- [cxxabi-forced.h](#)

5.8 `__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >` Struct Template Reference

Policy for shared `__pool` objects.

Inherits `__common_pool_base< _PoolTp, _Thread >`.

5.8.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread> struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
```

Policy for shared `__pool` objects.

Definition at line 456 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

5.9 `__gnu_cxx::detail::__mini_vector< _Tp >` Class Template Reference

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

Public Types

- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

5.10 `__gnu_cxx::__detail::_Bitmap_counter< _Tp >` Class Template Reference

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator __pos) throw ()
- void **insert** (iterator __pos, const_reference __x)
- reference **operator[]** (const size_type __pos) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference __x)
- size_type **size** () const throw ()

5.9.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::__mini_vector< _Tp >`

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`. It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses `operator new()` to get memory, and `operator delete()` to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 70 of file `bitmap_allocator.h`.

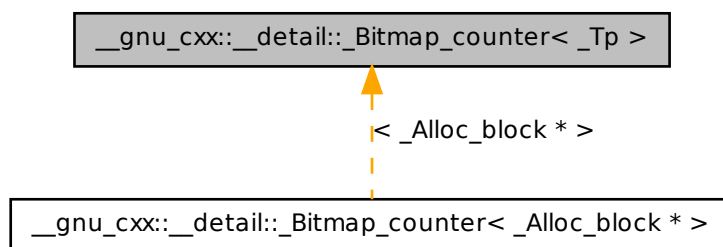
The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.10 `__gnu_cxx::__detail::_Bitmap_counter< _Tp >` Class Template Reference

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Inheritance diagram for `__gnu_cxx::__detail::_Bitmap_counter<_Tp>`:



Public Member Functions

- `_Bitmap_counter` ([_BPVector](#) &Rvbp, long __index=-1)
- `pointer _M_base` () const throw ()
- `bool _M_finished` () const throw ()
- `size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- `void _M_reset` (long __index=-1) throw ()
- `void _M_set_internal_bitmap` (size_t *__new_internal_marker) throw ()
- `_Index_type _M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

5.10.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_Bitmap_counter<_Tp>`

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Definition at line 399 of file `bitmap_allocator.h`.

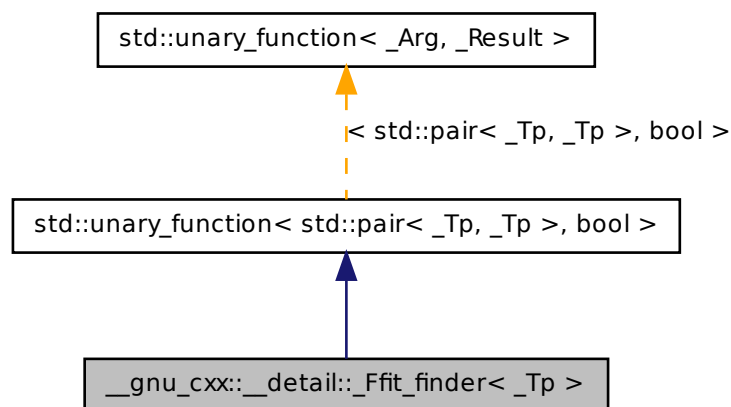
The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.11 `__gnu_cxx::__detail::_Ffit_finder< _Tp >` Class Template Reference

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder< _Tp >`:



Public Types

- typedef `std::pair< _Tp, _Tp >` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `size_t * _M_get () const throw ()`
- `_Counter_type _M_offset () const throw ()`
- `bool operator() (_Block_pair __bp) throw ()`

5.11.1 Detailed Description

template<typename _Tp> class __gnu_cxx::__detail::_Ffit_finder< _Tp >

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 334 of file `bitmap_allocator.h`.

5.11.2 Member Typedef Documentation

5.11.2.1 typedef std::pair< _Tp, _Tp > std::unary_function< std::pair< _Tp, _Tp >, bool >::argument_type [inherited]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.11.2.2 typedef bool std::unary_function< std::pair< _Tp, _Tp >, bool >::result_type [inherited]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

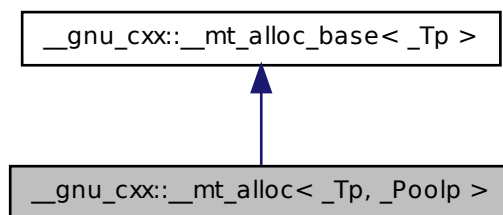
- [bitmap_allocator.h](#)

5.12 __gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `__mt_alloc` (const `__mt_alloc` &) throw ()
- `template<typename _Tp1, typename _Poolp1>`
`__mt_alloc` (const `__mt_alloc<_Tp1, _Poolp1>` &) throw ()
- `const __pool_base::_Tune _M_get_options` ()
- `void _M_set_options` (__pool_base::_Tune __t)
- `pointer address` (reference __x) const
- `const_pointer address` (const_reference __x) const
- `pointer allocate` (size_type __n, const void * = 0)
- `template<typename... _Args>`
`void construct` (pointer __p, _Args &&... __args)
- `void construct` (pointer __p, const _Tp & __val)
- `void deallocate` (pointer __p, size_type __n)

- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

5.12.1 Detailed Description

template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >> class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Definition at line 625 of file mt_allocator.h.

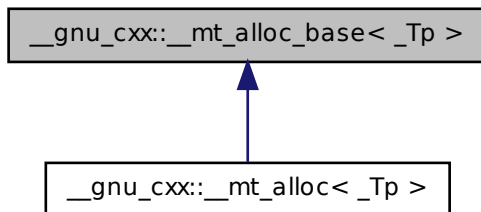
The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.13 __gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference

Base class for _Tp dependent member functions.

Inheritance diagram for __gnu_cxx::__mt_alloc_base< _Tp >:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`
- `template<typename... _Args>
void construct (pointer __p, _Args &&...__args)`
- `void construct (pointer __p, const _Tp &__val)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.13.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__mt_alloc_base< _Tp >`

Base class for `_Tp` dependent member functions.

Definition at line 566 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.14 `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct` Template Reference

Policy for individual `__pool` objects.

Inherits `__per_type_pool_base< _Tp, _PoolTp, _Thread >`.

5.14.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread> struct
__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >
```

Policy for individual `__pool` objects.

Definition at line 551 of file `mt_allocator.h`.

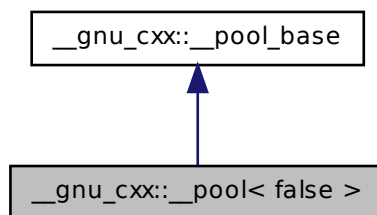
The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

5.15 `__gnu_cxx::__pool< false >` Class Template Reference

Specialization for single thread.

Inheritance diagram for `__gnu_cxx::__pool< false >`:



Public Types

- typedef unsigned short int `_Binmap_type`

Public Member Functions

- `__pool` (const `__pool_base::_Tune` & `_tune`)
- void `_M_adjust_freelist` (const `_Bin_record` &, `_Block_record` *, `size_t`)

- `bool _M_check_threshold (size_t __bytes)`
- `void _M_destroy () throw ()`
- `size_t _M_get_align ()`
- `const _Bin_record & _M_get_bin (size_t __which)`
- `size_t _M_get_binmap (size_t __bytes)`
- `const _Tune & _M_get_options () const`
- `size_t _M_get_thread_id ()`
- `void _M_initialize_once ()`
- `void _M_reclaim_block (char *__p, size_t __bytes) throw ()`
- `char * _M_reserve_block (size_t __bytes, const size_t __thread_id)`
- `void _M_set_options (_Tune __t)`

Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

5.15.1 Detailed Description

`template<> class __gnu_cxx::__pool< false >`

Specialization for single thread.

Definition at line 192 of file `mt_allocator.h`.

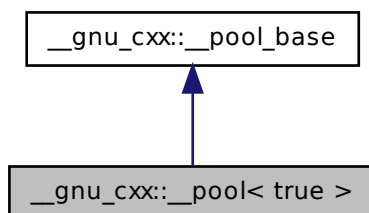
The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.16 `__gnu_cxx::__pool< true >` Class Template Reference

Specialization for thread enabled, via `gthreads.h`.

Inheritance diagram for `__gnu_cxx::__pool< true >`:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool** (const __pool_base::Tune &__tune)
- **__attribute__** ((__const__)) void **_M_destroy_thread_key**(void *) throw ()
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record *__block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

5.16.1 Detailed Description

`template<> class __gnu_cxx::__pool< true >`

Specialization for thread enabled, via `gthreads.h`.

Definition at line 259 of file `mt_allocator.h`.

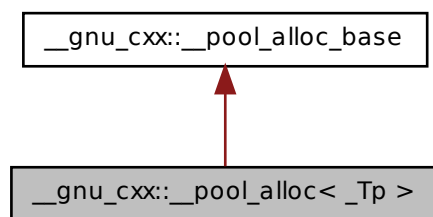
The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.17 `__gnu_cxx::__pool_alloc<_Tp>` Class Template Reference

Allocator using a memory pool with a single lock.

Inheritance diagram for `__gnu_cxx::__pool_alloc<_Tp>`:



Public Types

- `typedef const _Tp * const_pointer`

- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__pool_alloc** (const [__pool_alloc](#) &) throw ()
- template<typename _Tp1 >
 __pool_alloc (const [__pool_alloc](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, const void * = 0)
- template<typename... _Args>
 void **construct** (pointer __p, _Args &&... __args)
- void **construct** (pointer __p, const _Tp & __val)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

Private Types

- enum { **_S_align** }
- enum { **_S_max_bytes** }
- enum { **_S_free_list_size** }

Private Member Functions

- **__attribute__** ((__const__)) _Obj *volatile *_M_get_free_list(size_t __bytes) throw ()
- char * **_M_allocate_chunk** (size_t __n, int & __nobjs)
- __mutex & **_M_get_mutex** () throw ()
- void * **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Private Attributes

- static char * **_S_end_free**
- static _Obj *volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char * **_S_start_free**

5.17.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__pool_alloc< _Tp >`

Allocator using a memory pool with a single lock.

Definition at line 122 of file `pool_allocator.h`.

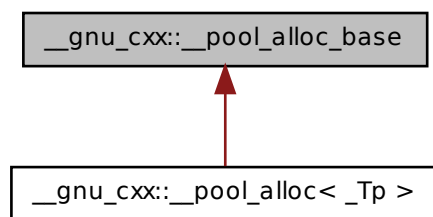
The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

5.18 __gnu_cxx::__pool_alloc_base Class Reference

Base class for [__pool_alloc](#).

Inheritance diagram for `__gnu_cxx::__pool_alloc_base`:



Protected Types

- enum { `_S_align` }
- enum { `_S_max_bytes` }
- enum { `_S_free_list_size` }

Protected Member Functions

- `__attribute__((__const__)) _Obj *volatile *_M_get_free_list(size_t __bytes) throw ()`

- `char * _M_allocate_chunk (size_t __n, int &__nobjs)`
- `__mutex & _M_get_mutex () throw ()`
- `void * _M_refill (size_t __n)`
- `size_t _M_round_up (size_t __bytes)`

Static Protected Attributes

- `static char * _S_end_free`
- `static _Obj *volatile _S_free_list [_S_free_list_size]`
- `static size_t _S_heap_size`
- `static char * _S_start_free`

5.18.1 Detailed Description

Base class for [__pool_alloc](#). Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size > `_S_max_bytes`, the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

Definition at line 74 of file `pool_allocator.h`.

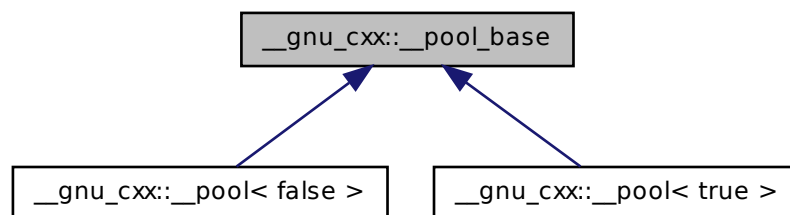
The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

5.19 `__gnu_cxx::__pool_base` Struct Reference

Base class for pool object.

Inheritance diagram for __gnu_cxx::__pool_base:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool_base** (const _Tune &__options)
- **bool _M_check_threshold** (size_t __bytes)
- **size_t _M_get_align** ()
- **size_t _M_get_binmap** (size_t __bytes)
- **const _Tune & _M_get_options** () const
- **void _M_set_options** (_Tune __t)

Protected Attributes

- **_Binmap_type * _M_binmap**
- **bool _M_init**
- **_Tune _M_options**

5.19.1 Detailed Description

Base class for pool object.

Definition at line 47 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

5.20 `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility< _CharT, _Traits, _Alloc >`.

Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility< _CharT, _Traits, _Alloc > _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

Public Member Functions

- `__rc_string_base (const _Alloc &__a)`
- `__rc_string_base (const __rc_string_base &__rcs)`
- `__rc_string_base (__rc_string_base &&__rcs)`
- `__rc_string_base (size_type __n, _CharT __c, const _Alloc &__a)`
- `template<typename _InputIterator >
__rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)`
- `void _M_assign (const __rc_string_base &__rcs)`
- `size_type _M_capacity () const`
- `void _M_clear ()`
- `bool _M_compare (const __rc_string_base &) const`
- `template<>
bool _M_compare (const __rc_string_base &__rcs) const`
- `template<>
bool _M_compare (const __rc_string_base &__rcs) const`
- `_CharT * _M_data () const`
- `void _M_erase (size_type __pos, size_type __n)`
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- `bool _M_is_shared () const`
- `void _M_leak ()`
- `size_type _M_length () const`

- `size_type _M_max_size () const`
- `void _M_mutate (size_type __pos, size_type __len1, const _CharT *__s, size_type __len2)`
- `void _M_reserve (size_type __res)`
- `void _M_set_leaked ()`
- `void _M_set_length (size_type __n)`
- `void _M_swap (__rc_string_base &__rcs)`
- `template<typename _InIterator >
_CharT * _S_construct (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward_iterator_tag)`

Protected Types

- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __rc_string_base>> __const_rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __sso_string_base>> __const_sso_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __rc_string_base>> __rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __sso_string_base>> __sso_iterator`
- `typedef _CharT_alloc_type::const_pointer const_pointer`
- `typedef _CharT_alloc_type::difference_type difference_type`
- `typedef _CharT_alloc_type::pointer pointer`

Static Protected Member Functions

- `static void _S_assign (_CharT *__d, size_type __n, _CharT __c)`
- `static int _S_compare (size_type __n1, size_type __n2)`
- `static void _S_copy (_CharT *__d, const _CharT *__s, size_type __n)`
- `static void _S_copy_chars (_CharT *__p, _CharT *__k1, _CharT *__k2)`
- `static void _S_copy_chars (_CharT *__p, __sso_iterator __k1, __sso_iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __rc_iterator __k1, __rc_iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2)`
- `template<typename _Iterator >
static void _S_copy_chars (_CharT *__p, _Iterator __k1, _Iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2)`

- static void `_S_copy_chars` (`_CharT *__p`, const `_CharT *__k1`, const `_CharT *__k2`)
- static void `_S_move` (`_CharT *__d`, const `_CharT *__s`, `size_type __n`)

5.20.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`

Documentation? What's that? Nathan Myers <ncm@cantrip.org>.

A string looks like this:

	<code>[_Rep]</code>
	<code>_M_length</code>
<code>[_rc_string_base<char_type>]</code>	<code>_M_capacity</code>
<code>_M_dataplus</code>	<code>_M_refcount</code>
<code>_M_p -----></code>	unnamed array of <code>char_type</code>

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc_string_base.h](#)

5.21 `__gnu_cxx::__scoped_lock` Class Reference

Scoped lock idiom.

Public Types

- typedef `__mutex` `__mutex_type`

Public Member Functions

- `__scoped_lock` (`__mutex_type` & `__name`)

5.21.1 Detailed Description

Scoped lock idiom.

Definition at line 241 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

5.22 `__gnu_cxx::__versa_string< _CharT, _Traits, _-Alloc, _Base >` Class Template Reference

Template class [__versa_string](#).

Data structure managing sequences of characters and character-like objects.

Inherits `_Base< _CharT, _Traits, _Alloc >`.

Public Types

- typedef `_Alloc` `allocator_type`
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` `const_iterator`
- typedef `_CharT_alloc_type::const_pointer` `const_pointer`
- typedef `const value_type &` `const_reference`
- typedef `std::reverse_iterator< const_iterator >` `const_reverse_iterator`
- typedef `_CharT_alloc_type::difference_type` `difference_type`
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` `iterator`
- typedef `_CharT_alloc_type::pointer` `pointer`

- typedef value_type & **reference**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- [__versa_string](#) ()
- [__versa_string](#) (const _Alloc &__a)
- [__versa_string](#) ([__versa_string](#) &&__str)
- [__versa_string](#) (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- [__versa_string](#) (const _CharT *__s, const _Alloc &__a=_Alloc())
- [__versa_string](#) ([std::initializer_list](#)< _CharT > __l, const _Alloc &__a=_Alloc())
- [__versa_string](#) (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- template<class _InputIterator >
 [__versa_string](#) (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- [__versa_string](#) (const [__versa_string](#) &__str)
- [__versa_string](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n=npos)
- [__versa_string](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n, const _Alloc &__a)
- [~__versa_string](#) ()
- [__versa_string](#) & [append](#) (const [__versa_string](#) &__str)
- [__versa_string](#) & [append](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n)
- [__versa_string](#) & [append](#) (const _CharT *__s, size_type __n)
- [__versa_string](#) & [append](#) (const _CharT *__s)
- [__versa_string](#) & [append](#) (size_type __n, _CharT __c)
- [__versa_string](#) & [append](#) ([std::initializer_list](#)< _CharT > __l)
- template<class _InputIterator >
 [__versa_string](#) & [append](#) (_InputIterator __first, _InputIterator __last)
- [__versa_string](#) & [assign](#) (const _CharT *__s)
- [__versa_string](#) & [assign](#) (size_type __n, _CharT __c)
- template<class _InputIterator >
 [__versa_string](#) & [assign](#) (_InputIterator __first, _InputIterator __last)
- [__versa_string](#) & [assign](#) ([std::initializer_list](#)< _CharT > __l)
- [__versa_string](#) & [assign](#) (const [__versa_string](#) &__str)
- [__versa_string](#) & [assign](#) ([__versa_string](#) &&__str)
- [__versa_string](#) & [assign](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n)

- `__versa_string` & `assign` (const `_CharT` *`__s`, `size_type` `__n`)
- `const_reference` `at` (`size_type` `__n`) const
- `reference` `at` (`size_type` `__n`)
- `reference` `back` ()
- `const_reference` `back` () const
- `iterator` `begin` ()
- `const_iterator` `begin` () const
- `const _CharT` * `c_str` () const
- `size_type` `capacity` () const
- `const_iterator` `cbegin` () const
- `const_iterator` `cend` () const
- `void` `clear` ()
- `int` `compare` (const `_CharT` *`__s`) const
- `int` `compare` (`size_type` `__pos`, `size_type` `__n1`, const `_CharT` *`__s`) const
- `int` `compare` (`size_type` `__pos`, `size_type` `__n1`, const `_CharT` *`__s`, `size_type` `__n2`) const
- `int` `compare` (`size_type` `__pos1`, `size_type` `__n1`, const `__versa_string` &`__str`, `size_type` `__pos2`, `size_type` `__n2`) const
- `int` `compare` (`size_type` `__pos`, `size_type` `__n`, const `__versa_string` &`__str`) const
- `int` `compare` (const `__versa_string` &`__str`) const
- `size_type` `copy` (`_CharT` *`__s`, `size_type` `__n`, `size_type` `__pos=0`) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- `const _CharT` * `data` () const
- `bool` `empty` () const
- `iterator` `end` ()
- `const_iterator` `end` () const
- `__versa_string` & `erase` (`size_type` `__pos=0`, `size_type` `__n=npos`)
- `iterator` `erase` (`iterator` `__position`)
- `iterator` `erase` (`iterator` `__first`, `iterator` `__last`)
- `size_type` `find` (`_CharT` `__c`, `size_type` `__pos=0`) const
- `size_type` `find` (const `_CharT` *`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type` `find` (const `__versa_string` &`__str`, `size_type` `__pos=0`) const
- `size_type` `find` (const `_CharT` *`__s`, `size_type` `__pos=0`) const
- `size_type` `find_first_not_of` (const `_CharT` *`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type` `find_first_not_of` (`_CharT` `__c`, `size_type` `__pos=0`) const
- `size_type` `find_first_not_of` (const `__versa_string` &`__str`, `size_type` `__pos=0`) const
- `size_type` `find_first_not_of` (const `_CharT` *`__s`, `size_type` `__pos=0`) const
- `size_type` `find_first_of` (`_CharT` `__c`, `size_type` `__pos=0`) const
- `size_type` `find_first_of` (const `_CharT` *`__s`, `size_type` `__pos`, `size_type` `__n`) const

- size_type [find_first_of](#) (const _CharT *__s, size_type __pos=0) const
- size_type [find_first_of](#) (const __versa_string &__str, size_type __pos=0) const
- size_type [find_last_not_of](#) (_CharT __c, size_type __pos=npos) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_last_not_of](#) (const __versa_string &__str, size_type __pos=npos) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos=npos) const
- size_type [find_last_of](#) (_CharT __c, size_type __pos=npos) const
- size_type [find_last_of](#) (const __versa_string &__str, size_type __pos=npos) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos=npos) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const
- void [insert](#) (iterator __p, size_type __n, _CharT __c)
- template<class _InputIterator >
void [insert](#) (iterator __p, _InputIterator __beg, _InputIterator __end)
- void [insert](#) (iterator __p, [std::initializer_list](#)< _CharT > __l)
- __versa_string & [insert](#) (size_type __pos1, const __versa_string &__str, size_type __pos2, size_type __n)
- __versa_string & [insert](#) (size_type __pos, const _CharT *__s, size_type __n)
- __versa_string & [insert](#) (size_type __pos, const _CharT *__s)
- iterator [insert](#) (iterator __p, _CharT __c)
- __versa_string & [insert](#) (size_type __pos, size_type __n, _CharT __c)
- __versa_string & [insert](#) (size_type __pos1, const __versa_string &__str)
- size_type [length](#) () const
- size_type [max_size](#) () const
- __versa_string & [operator+=](#) ([std::initializer_list](#)< _CharT > __l)
- __versa_string & [operator+=](#) (const __versa_string &__str)
- __versa_string & [operator+=](#) (_CharT __c)
- __versa_string & [operator+=](#) (const _CharT *__s)
- __versa_string & [operator=](#) (__versa_string &&__str)
- __versa_string & [operator=](#) (const _CharT *__s)
- __versa_string & [operator=](#) (_CharT __c)
- __versa_string & [operator=](#) ([std::initializer_list](#)< _CharT > __l)
- __versa_string & [operator=](#) (const __versa_string &__str)
- const_reference [operator\[\]](#) (size_type __pos) const
- reference [operator\[\]](#) (size_type __pos)
- void [push_back](#) (_CharT __c)
- reverse_iterator [rbegin](#) ()

- `const_reverse_iterator rbegin` () const
- `reverse_iterator rend` ()
- `const_reverse_iterator rend` () const
- `__versa_string & replace` (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- `__versa_string & replace` (iterator __i1, iterator __i2, const __versa_string &__str)
- `__versa_string & replace` (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- `__versa_string & replace` (size_type __pos, size_type __n1, const _CharT *__s)
- `__versa_string & replace` (size_type __pos, size_type __n, const __versa_string &__str)
- `__versa_string & replace` (iterator __i1, iterator __i2, `std::initializer_list<_CharT>` __l)
- `template<class _InputIterator>`
 `__versa_string & replace` (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, const _CharT *__s)
- `__versa_string & replace` (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- `__versa_string & replace` (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)
- `__versa_string & replace` (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- `__versa_string & replace` (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- `void reserve` (size_type __res_arg=0)
- `void resize` (size_type __n)
- `void resize` (size_type __n, _CharT __c)
- `size_type rfind` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type rfind` (const __versa_string &__str, size_type __pos=`npos`) const
- `size_type rfind` (const _CharT *__s, size_type __pos=`npos`) const
- `size_type rfind` (_CharT __c, size_type __pos=`npos`) const
- `void shrink_to_fit` ()
- `size_type size` () const
- `__versa_string substr` (size_type __pos=0, size_type __n=`npos`) const
- `void swap` (__versa_string &__s)

Static Public Attributes

- static const size_type `npos`

5.22.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template<
typename, typename, typename > class _Base> class __gnu_cxx::__versa_
string< _CharT, _Traits, _Alloc, _Base >
```

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

Definition at line 52 of file `vstring.h`.

5.22.2 Constructor & Destructor Documentation

```
5.22.2.1 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( ) [inline]
```

Default constructor creates an empty string.

Definition at line 130 of file `vstring.h`.

```
5.22.2.2 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( const _Alloc & __a ) [inline, explicit]
```

Construct an empty string using allocator *a*.

Definition at line 137 of file `vstring.h`.

```
5.22.2.3 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str ) [inline]
```

Construct string with copy of value of *str*.

Parameters

`__str` Source string.

Definition at line 145 of file `vstring.h`.

```
5.22.2.4 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( __versa_string< _CharT, _Traits, _Alloc, _Base >
&& __str ) [inline]
```

String move constructor.

Parameters

`__str` Source string.

The newly-constructed string contains the exact contents of `str`. The contents of `str` are a valid, but unspecified string.

Definition at line 157 of file `vstring.h`.

```
5.22.2.5 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( std::initializer_list< _CharT > __l, const _Alloc
& __a = _Alloc() ) [inline]
```

Construct string from an initializer list.

Parameters

`__l` `std::initializer_list` of characters.

`__a` Allocator to use (default is default allocator).

Definition at line 165 of file `vstring.h`.

```
5.22.2.6 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos, size_type __n = npos )
[inline]
```

Construct string as copy of a substring.

Parameters

- `__str` Source string.
- `__pos` Index of first character to copy from.
- `__n` Number of characters to copy (default remainder).

Definition at line 176 of file `vstring.h`.

```
5.22.2.7  template<typename _CharT, typename _Traits, typename _Alloc,
           template< typename, typename, typename > class _Base>
           __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
           >::__versa_string ( const __versa_string< _CharT, _Traits, _Alloc,
           _Base > & __str, size_type __pos, size_type __n, const _Alloc & __a
           ) [inline]
```

Construct string as copy of a substring.

Parameters

- `__str` Source string.
- `__pos` Index of first character to copy from.
- `__n` Number of characters to copy.
- `__a` Allocator to use.

Definition at line 191 of file `vstring.h`.

```
5.22.2.8  template<typename _CharT, typename _Traits, typename _Alloc,
           template< typename, typename, typename > class _Base>
           __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
           >::__versa_string ( const _CharT * __s, size_type __n, const _Alloc
           & __a = _Alloc() ) [inline]
```

Construct string initialized by a character array.

Parameters

- `__s` Source character array.
- `__n` Number of characters to copy.
- `__a` Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 208 of file `vstring.h`.

5.22.2.9 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::__versa_string (const _CharT * __s, const _Alloc & __a =
_Alloc()) [inline]`

Construct string as copy of a C string.

Parameters

`__s` Source C string.

`__a` Allocator to use (default is default allocator).

Definition at line 217 of file `vstring.h`.

5.22.2.10 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::__versa_string (size_type __n, _CharT __c, const _Alloc & __a
= _Alloc()) [inline]`

Construct string as multiple characters.

Parameters

`__n` Number of characters.

`__c` Character to use.

`__a` Allocator to use (default is default allocator).

Definition at line 227 of file `vstring.h`.

5.22.2.11 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
template<class _InputIterator > __gnu_cxx::__versa_string<
_CharT, _Traits, _Alloc, _Base >::__versa_string (_InputIterator
__beg, _InputIterator __end, const _Alloc & __a = _Alloc())
[inline]`

Construct string as copy of a range.

Parameters

`__beg` Start of range.

`__end` End of range.

`__a` Allocator to use (default is default allocator).

Definition at line 237 of file `vstring.h`.

```
5.22.2.12  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::~__versa_string( ) [inline]
```

Destroy the string instance.

Definition at line 244 of file `vstring.h`.

5.22.3 Member Function Documentation

```
5.22.3.1  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const __versa_string< _CharT, _Traits, _Alloc,
            _Base > & __str ) [inline]
```

Append a string to this string.

Parameters

`__str` The string to append.

Returns

Reference to this string.

Definition at line 676 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `std::getline()`, and `__gnu_cxx::operator+()`.

```
5.22.3.2  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const __versa_string< _CharT, _Traits, _Alloc,
            _Base > & __str, size_type __pos, size_type __n ) [inline]
```

Append a substring.

Parameters

- `__str` The string to append.
- `__pos` Index of the first character of `str` to append.
- `__n` The number of characters to append.

Returns

Reference to this string.

Exceptions

[`std::out_of_range`](#) if `pos` is not a valid index.

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 693 of file `vstring.h`.

```
5.22.3.3  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const _CharT * __s, size_type __n ) [inline]
```

Append a C substring.

Parameters

- `__s` The C string to append.
- `__n` The number of characters to append.

Returns

Reference to this string.

Definition at line 705 of file `vstring.h`.

```
5.22.3.4  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const _CharT * __s ) [inline]
```

Append a C string.

Parameters

`__s` The C string to append.

Returns

Reference to this string.

Definition at line 718 of file `vstring.h`.

5.22.3.5 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append (size_type __n, _CharT __c) [inline]`

Append multiple characters.

Parameters

`__n` The number of characters to append.

`__c` The character to use.

Returns

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 735 of file `vstring.h`.

5.22.3.6 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append (std::initializer_list< _CharT > __l) [inline]`

Append an `initializer_list` of characters.

Parameters

`__l` The `initializer_list` of characters to append.

Returns

Reference to this string.

Definition at line 745 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

5.22.3.7 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> template<class _InputIterator > __versa_string&
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::append (_InputIterator __first, _InputIterator __last)
[inline]`

Append a range of characters.

Parameters

`__first` Iterator referencing the first character to append.
`__last` Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 759 of file `vstring.h`.

5.22.3.8 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::assign (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str) [inline]`

Set value to contents of another string.

Parameters

`__str` Source string to use.

Returns

Reference to this string.

Definition at line 782 of file `vstring.h`.

5.22.3.9 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::assign (__versa_string< _CharT, _Traits, _Alloc, _Base >
&& __str) [inline]`

Set value to contents of another string.

Parameters

`__str` Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 798 of file `vstring.h`.

```
5.22.3.10  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::assign ( const __versa_string< _CharT, _Traits,
            _Alloc, _Base > & __str, size_type __pos, size_type __n )
            [inline]
```

Set value to a substring of a string.

Parameters

`__str` The string to use.

`__pos` Index of the first character of str.

`__n` Number of characters to use.

Returns

Reference to this string.

Exceptions

[*`std::out_of_range`*](#) if `__pos` is not a valid index.

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 819 of file `vstring.h`.

```
5.22.3.11  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::assign ( const _CharT * __s, size_type __n )
            [inline]
```

Set value to a C substring.

Parameters

- `__s` The C string to use.
- `__n` Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 836 of file `vstring.h`.

5.22.3.12 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::assign (const _CharT * __s) [inline]`

Set value to contents of a C string.

Parameters

- `__s` The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 852 of file `vstring.h`.

5.22.3.13 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::assign (size_type __n, _CharT __c) [inline]`

Set value to multiple characters.

Parameters

- `__n` Length of the resulting string.
- `__c` The character to use.

Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 869 of file `vstring.h`.

5.22.3.14 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (_InputIterator __first, _InputIterator __last) [inline]`

Set value to a range of characters.

Parameters

`__first` Iterator referencing the first character to append.

`__last` Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 883 of file `vstring.h`.

5.22.3.15 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (std::initializer_list< _CharT > __l) [inline]`

Set value to an `initializer_list` of characters.

Parameters

`__l` The `initializer_list` of characters to assign.

Returns

Reference to this string.

5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference 803

Definition at line 893 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

5.22.3.16 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at (size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

`__n` The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

[*`std::out_of_range`*](#) If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 567 of file `vstring.h`.

5.22.3.17 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at (size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

`__n` The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

[*`std::out_of_range`*](#) If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 586 of file `vstring.h`.

5.22.3.18 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::back () const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 624 of file `vstring.h`.

5.22.3.19 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> reference
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back
() [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 616 of file `vstring.h`.

5.22.3.20 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin
() [inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 310 of file `vstring.h`.

5.22.3.21 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 321 of file `vstring.h`.

5.22.3.22 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> const
_CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::c_str () const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1485 of file `vstring.h`.

5.22.3.23 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::capacity () const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 478 of file `vstring.h`.

5.22.3.24 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 385 of file `vstring.h`.

5.22.3.25 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 393 of file `vstring.h`.

5.22.3.26 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear
() [inline]`

Erases the string, making it empty.

Definition at line 506 of file `vstring.h`.

5.22.3.27 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::compare(const __versa_string< _CharT, _Traits, _Alloc, _Base
> & __str) const [inline]`

Compare to a string.

Parameters

`__str` String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1904 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

5.22.3.28 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::compare(size_type __pos, size_type __n, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __str) const`

Compare substring to a string.

Parameters

- `__pos` Index of first character of substring.
- `__n` Number of characters in substring.
- `__str` String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 458 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.29 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

- `__pos1` Index of first character of substring.
- `__n1` Number of characters in substring.
- `__str` String to compare against.
- `__pos2` Index of first character of substring of `str`.
- `__n2` Number of characters in substring of `str`.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer `< 0` if this substring is ordered before the substring of `__str`, `0` if their values are equivalent, or `> 0` if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 475 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `std::min()`.

5.22.3.30 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (const _CharT * __s) const`

Compare to a C string.

Parameters

`__s` C string to compare against.

Returns

Integer `< 0`, `0`, or `> 0`.

Returns an integer `< 0` if this string is ordered before `__s`, `0` if their values are equivalent, or `> 0` if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 494 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.31 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::compare (size_type __pos, size_type __n1, const _CharT * __s
) const`

Compare substring to a C string.

Parameters

`__pos` Index of first character of substring.
`__n1` Number of characters in substring.
`__s` C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 510 of file `vstring.tcc`.

References `std::min()`.

5.22.3.32 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::compare (size_type __pos, size_type __n1, const _CharT * __s,
size_type __n2) const`

Compare substring against a character array.

Parameters

`__pos1` Index of first character of substring.
`__n1` Number of characters in substring.
`__s` character array to compare against.
`__n2` Number of characters of `s`.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `\0` has no special meaning.

Definition at line 527 of file `vstring.tcc`.

References `std::min()`.

```
5.22.3.33  template<typename _CharT, typename _Traits, typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::copy
            ( _CharT * __s, size_type __n, size_type __pos = 0 ) const
```

Copy substring into C string.

Parameters

`__s` C string to copy value into.

`__n` Number of characters to copy.

`__pos` Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

[`std::out_of_range`](#) If `pos > size()`.

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 253 of file `vstring.tcc`.

5.22.3.34 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::crbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 402 of file `vstring.h`.

5.22.3.35 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 411 of file `vstring.h`.

5.22.3.36 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> const
_CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::data () const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1495 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

5.22.3.37 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> bool
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::empty () const [inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 514 of file `vstring.h`.

5.22.3.38 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 340 of file `vstring.h`.

5.22.3.39 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () [inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 329 of file `vstring.h`.

5.22.3.40 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::erase (size_type __pos = 0, size_type __n = npos)
[inline]`

Remove characters.

Parameters

`__pos` Index of first character to remove (default 0).

`__n` Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos` is beyond the end of this string.

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1088 of file `vstring.h`.

Referenced by `std::getline()`.

5.22.3.41 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> iterator
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase
 (iterator __position) [inline]`

Remove one character.

Parameters

`__position` Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1104 of file `vstring.h`.

5.22.3.42 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> iterator
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase
 (iterator __first, iterator __last) [inline]`

Remove a range of characters.

Parameters

`__first` Iterator referencing the first character to remove.

`__last` Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1125 of file `vstring.h`.

5.22.3.43 `template<typename _CharT, typename _Traits , typename _Alloc
 , template< typename, typename, typename > class _Base>
 __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (
 const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a C substring.

Parameters

`__s` C string to locate.
`__pos` Index of character to search from.
`__n` Number of characters from `__s` to search for.

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 268 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

5.22.3.44 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0) const [inline]`

Find position of a string.

Parameters

`__str` String to locate.
`__pos` Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1531 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

5.22.3.45 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find(
 const _CharT* __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

`__s` C string to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1545 of file `vstring.h`.

5.22.3.46 `template<typename _CharT, typename _Traits, typename _Alloc
 , template< typename, typename, typename > class _Base>
 __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find(
 _CharT __c, size_type __pos = 0) const`

Find position of a character.

Parameters

`__c` Character to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 292 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.47 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_first_not_of (const _CharT * __s, size_type __pos,
size_type __n) const`

Find position of a character not in C substring.

Parameters

- `__s` C string containing characters to avoid.
- `__pos` Index of character to search from.
- `__n` Number of characters from `s` to consider.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 390 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.48 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_first_not_of (_CharT __c, size_type __pos = 0) const`

Find position of a different character.

Parameters

- `__c` Character to avoid.
- `__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 403 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.49 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> size_type`
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`
`>::find_first_not_of (const __versa_string<_CharT, _Traits, _Alloc,`
`_Base > & __str, size_type __pos = 0) const [inline]`

Find position of a character not in string.

Parameters

`__str` String containing characters to avoid.
`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1759 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, and
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

5.22.3.50 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> size_type`
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`
`>::find_first_not_of (const _CharT * __s, size_type __pos = 0)`
`const [inline]`

Find position of a character not in C string.

Parameters

`__s` C string containing characters to avoid.
`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1789 of file `vstring.h`.

```
5.22.3.51  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base> size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_first_of( const __versa_string< _CharT, _Traits, _Alloc,
            _Base > & __str, size_type __pos = 0 ) const  [inline]
```

Find position of a character of string.

Parameters

`__str` String containing characters to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1634 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`.

```
5.22.3.52  template<typename _CharT, typename _Traits, typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_first_of( const _CharT * __s, size_type __pos, size_type
            __n ) const
```

Find position of a character of C substring.

Parameters

- `__s` String containing characters to locate.
- `__pos` Index of character to search from.
- `__n` Number of characters from `s` to search for.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 351 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.53 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

- `__s` String containing characters to locate.
- `__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1663 of file `vstring.h`.

5.22.3.54 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of (_CharT __c, size_type __pos = 0) const [inline]`

Find position of a character.

Parameters

`__c` Character to locate.
`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1682 of file `vstring.h`.

```
5.22.3.55  template<typename _CharT, typename _Traits, typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_last_not_of ( _CharT __c, size_type __pos = npos ) const
```

Find last position of a different character.

Parameters

`__c` Character to avoid.
`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 437 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.56  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base> size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_last_not_of ( const __versa_string< _CharT, _Traits, _Alloc,
            _Base > & __str, size_type __pos = npos ) const [inline]
```

Find last position of a character not in string.

Parameters

`__str` String containing characters to avoid.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1820 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

5.22.3.57 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character not in C substring.

Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search back from.

`__n` Number of characters from `s` to consider.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 415 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.58 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::find_last_not_of (const _CharT * __s, size_type __pos = npos)
 const [inline]`

Find last position of a character not in C string.

Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1851 of file `vstring.h`.

5.22.3.59 `template<typename _CharT, typename _Traits , typename _Alloc
 , template< typename, typename, typename > class _Base>
 __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::find_last_of (const _CharT * __s, size_type __pos, size_type
 __n) const`

Find last position of a character of C substring.

Parameters

`__s` C string containing characters to locate.

`__pos` Index of character to search back from.

`__n` Number of characters from `s` to search for.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 368 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.60 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::find_last_of (const __versa_string< _CharT, _Traits, _Alloc,
 _Base > & __str, size_type __pos = npos) const [inline]`

Find last position of a character of string.

Parameters

`__str` String containing characters to locate.
`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1697 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

5.22.3.61 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::find_last_of (const _CharT * __s, size_type __pos = npos)
 const [inline]`

Find last position of a character of C string.

Parameters

`__s` C string containing characters to locate.
`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1726 of file `vstring.h`.

5.22.3.62 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_of (_CharT __c, size_type __pos = npos) const
[inline]`

Find last position of a character.

Parameters

`__c` Character to locate.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1745 of file `vstring.h`.

5.22.3.63 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> reference
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front
() [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 600 of file `vstring.h`.

5.22.3.64 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::front () const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 608 of file `vstring.h`.

5.22.3.65 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> allocator_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::get_allocator () const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1502 of file `vstring.h`.

5.22.3.66 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, const _CharT * __s, size_type __n) [inline]`

Insert a C substring.

Parameters

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

`__n` The number of characters to insert.

Returns

Reference to this string.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

[`std::out_of_range`](#) If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1002 of file `vstring.h`.

5.22.3.67 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (iterator __p, _CharT __c) [inline]`

Insert one character.

Parameters

`__p` Iterator referencing position in string to insert at.
`__c` The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1063 of file `vstring.h`.

5.22.3.68 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::insert (size_type __pos1, const __versa_string<`
`_CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Insert value of a string.

Parameters

`__pos1` Iterator referencing location in string to insert at.
`__str` The string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 956 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.69 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert
(iterator __p, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

`__p` Iterator referencing location in string to insert at.

`__n` Number of characters to insert

`__c` The character to insert.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 911 of file `vstring.h`.

5.22.3.70 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert
(iterator __p, std::initializer_list<_CharT> __l) [inline]`

Insert an `initializer_list` of characters.

Parameters

`__p` Iterator referencing location in string to insert at.

`__l` The `initializer_list` of characters to insert.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Definition at line 939 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

5.22.3.71 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
template<class _InputIterator > void __gnu_cxx::__versa_string<
_CharT, _Traits, _Alloc, _Base >::insert (iterator __p,
_InputIterator __beg, _InputIterator __end) [inline]`

Insert a range of characters.

Parameters

`__p` Iterator referencing location in string to insert at.

`__beg` Start of range.

`__end` End of range.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 928 of file `vstring.h`.

5.22.3.72 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::insert (size_type __pos, size_type __n, _CharT
__c) [inline]`

Insert multiple characters.

Parameters

`__pos` Index in string to insert at.

`__n` Number of characters to insert

`__c` The character to insert.

Returns

Reference to this string.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

[`std::out_of_range`](#) If `__pos` is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1045 of file `vstring.h`.

```
5.22.3.73  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::insert ( size_type __pos1, const __versa_string<
            _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2,
            size_type __n ) [inline]
```

Insert a substring.

Parameters

- `__pos1` Iterator referencing location in string to insert at.
- `__str` The string to insert.
- `__pos2` Start of characters in `str` to insert.
- `__n` Number of characters to insert.

Returns

Reference to this string.

Exceptions

- `std::length_error` If new length exceeds `max_size()`.
- `std::out_of_range` If `__pos1 > size()` or `__pos2 > __str.size()`.

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 979 of file `vstring.h`.

```
5.22.3.74  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::insert ( size_type __pos, const _CharT * __s )
            [inline]
```

Insert a C string.

Parameters

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

Returns

Reference to this string.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

[`std::out_of_range`](#) If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1021 of file `vstring.h`.

5.22.3.75 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::length () const [inline]`

Returns the number of characters in the string, not including any `///` null-termination.

Definition at line 426 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::compare()`.

5.22.3.76 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::max_size () const [inline]`

Returns the `size()` of the largest possible string.

Definition at line 431 of file `vstring.h`.

Referenced by `std::getline()`.

5.22.3.77 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::operator+=(const __versa_string<_CharT,
_Traits, _Alloc, _Base> & __str) [inline]`

Append a string to this string.

Parameters

`__str` The string to append.

Returns

Reference to this string.

Definition at line 635 of file `vstring.h`.

5.22.3.78 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::operator+=(const _CharT * __s) [inline]`

Append a C string.

Parameters

`__s` The C string to append.

Returns

Reference to this string.

Definition at line 644 of file `vstring.h`.

5.22.3.79 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::operator+=(_CharT __c) [inline]`

Append a character.

Parameters

`__c` The character to append.

Returns

Reference to this string.

Definition at line 653 of file vstring.h.

```
5.22.3.80  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::operator+=( std::initializer_list< _CharT > __l )
            [inline]
```

Append an initializer_list of characters.

Parameters

__l The initializer_list of characters to be appended.

Returns

Reference to this string.

Definition at line 666 of file vstring.h.

```
5.22.3.81  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::operator=( const _CharT * __s ) [inline]
```

Copy contents of __s into this string.

Parameters

__s Source null-terminated string.

Definition at line 287 of file vstring.h.

```
5.22.3.82  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::operator=( std::initializer_list< _CharT > __l )
            [inline]
```

Set value to string constructed from initializer list.

Parameters

`__l` [std::initializer_list](#).

Definition at line 275 of file `vstring.h`.

5.22.3.83 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::operator=(__versa_string<_CharT, _Traits,
_Alloc, _Base> && __str) [inline]`

String move assignment operator.

Parameters

`__str` Source string.

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 263 of file `vstring.h`.

5.22.3.84 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::operator=(_CharT __c) [inline]`

Set value to string of length 1.

Parameters

`__c` Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 298 of file `vstring.h`.

5.22.3.85 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::operator=(const __versa_string<_CharT, _Traits,
_Alloc, _Base> & __str) [inline]`

Assign the value of `str` to this string.

Parameters

`__str` Source string.

Definition at line 251 of file `vstring.h`.

5.22.3.86 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::operator[] (size_type __pos) const [inline]`

Subscript access to the data contained in the string.

Parameters

`__pos` The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 529 of file `vstring.h`.

5.22.3.87 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> reference
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::operator[] (size_type __pos) [inline]`

Subscript access to the data contained in the string.

Parameters

`__pos` The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see [at\(\)](#).) Unshares the string.

Definition at line 546 of file `vstring.h`.

5.22.3.88 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> void
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
 >::push_back (_CharT __c) [inline]`

Append a single character.

Parameters

`__c` Character to append.

Definition at line 767 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

5.22.3.89 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 349 of file `vstring.h`.

5.22.3.90 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 358 of file `vstring.h`.

5.22.3.91 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 376 of file `vstring.h`.

5.22.3.92 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 367 of file vstring.h.

5.22.3.93 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT,
_Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2,
std::initializer_list< _CharT > __l) [inline]`

Replace range of characters with initializer_list.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__l` The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1421 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

5.22.3.94 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (iterator __i1, iterator __i2, const
__versa_string<_CharT, _Traits, _Alloc, _Base> & __str)
[inline]`

Replace range of characters with string.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__str` String value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1270 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`,
and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

5.22.3.95 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (size_type __pos, size_type __n, const
__versa_string<_CharT, _Traits, _Alloc, _Base> & __str)
[inline]`

Replace characters with value from another string.

Parameters

`__pos` Index of first character to replace.

`__n` Number of characters to be replaced.

`__str` String to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos` is beyond the end of this string.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1153 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

5.22.3.96 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c) [inline]`

Replace characters with multiple characters.

Parameters

`__pos` Index of first character to replace.

`__n1` Number of characters to be replaced.

`__n2` Number of characters to insert.

`__c` Character to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos > size()`.

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1252 of file `vstring.h`.

5.22.3.97 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::replace (iterator __i1, iterator __i2, size_type
 __n, _CharT __c) [inline]`

Replace range of characters with multiple characters.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__n` Number of characters to insert.

`__c` Character to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1, i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1330 of file `vstring.h`.

5.22.3.98 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::replace (iterator __i1, iterator __i2, const
 _CharT* __s) [inline]`

Replace range of characters with C string.

Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__s` C string value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1309 of file `vstring.h`.

5.22.3.99 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2) [inline]`

Replace range of characters with range.

Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__k1` Iterator referencing start of range to insert.
- `__k2` Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1353 of file `vstring.h`.

5.22.3.100 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
 _Alloc, _Base >::replace (size_type __pos, size_type __n1, const
 _CharT * __s, size_type __n2) [inline]`

Replace characters with value of a C substring.

Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__s` C string to insert.
- `__n2` Number of characters from `__s` to use.

Returns

Reference to this string.

Exceptions

- [*std::out_of_range*](#) If `__pos1 > size()`.
- [*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1204 of file `vstring.h`.

5.22.3.101 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
 _Alloc, _Base >::replace (size_type __pos, size_type __n1, const
 _CharT * __s) [inline]`

Replace characters with value of a C string.

Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__s` C string to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos > size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1228 of file `vstring.h`.

```
5.22.3.102 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::replace ( size_type __pos1, size_type __n1,
const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
size_type __pos2, size_type __n2 ) [inline]
```

Replace characters with value from another string.

Parameters

`__pos1` Index of first character to replace.

`__n1` Number of characters to be replaced.

`__str` String to insert.

`__pos2` Index of first character of `str` to use.

`__n2` Number of characters from `str` to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos1 > size()` or `__pos2 > str.size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1176 of file `vstring.h`.

5.22.3.103 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (iterator __i1, iterator __i2, const
_CharT * __s, size_type __n) [inline]`

Replace range of characters with C substring.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__s` C string value to insert.

`__n` Number of characters from `s` to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1288 of file `vstring.h`.

5.22.3.104 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::reserve (size_type __res_arg = 0) [inline]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

`__res_arg` Number of characters required.

Exceptions

[std::length_error](#) If `__res_arg` exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 499 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

5.22.3.105 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::resize (size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

`__n` Number of characters the string should contain.

`__c` Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 49 of file `vstring.tcc`.

5.22.3.106 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::resize (size_type __n) [inline]`

Resizes the string to the specified number of characters.

Parameters

`__n` Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 458 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

5.22.3.107 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::rfind (const _CharT * __s, size_type __pos, size_type __n)
const`

Find last position of a C substring.

Parameters

`__s` C string to locate.
`__pos` Index of character to search back from.
`__n` Number of characters from `s` to search for.

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 311 of file `vstring.tcc`.

References `std::min()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.108 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::rfind (_CharT __c, size_type __pos = npos) const`

Find last position of a character.

Parameters

`__c` Character to locate.
`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 333 of file vstring.tcc.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.109  template<typename _CharT, typename _Traits, typename _Alloc,
               template< typename, typename, typename > class _Base> size_type
               __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
               >::rfind ( const _CharT * __s, size_type __pos = npos ) const
               [inline]
```

Find last position of a C string.

Parameters

`__s` C string to locate.

`__pos` Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1604 of file vstring.h.

```
5.22.3.110  template<typename _CharT, typename _Traits, typename _Alloc,
               template< typename, typename, typename > class _Base> size_type
               __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
               >::rfind ( const __versa_string< _CharT, _Traits, _Alloc, _Base >
               & __str, size_type __pos = npos ) const  [inline]
```

Find last position of a string.

Parameters

`__str` String to locate.

`__pos` Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

847

Definition at line 1575 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

5.22.3.111 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> void`
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`
`>::shrink_to_fit() [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 464 of file `vstring.h`.

5.22.3.112 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> size_type`
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size`
`() const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 420 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

5.22.3.113 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string __gnu_cxx::__versa_string<_CharT, _Traits,`
`_Alloc, _Base>::substr(size_type __pos = 0, size_type __n = npos`
`) const [inline]`

Get a substring.

Parameters

`__pos` Index of first character (default 0).
`__n` Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

[`std::out_of_range`](#) If `pos > size()`.

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 1883 of file `vstring.h`.

5.22.3.114 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> void
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & __s
) [inline]`

Swap contents with another string.

Parameters

`__s` String to swap with.

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1474 of file `vstring.h`.

Referenced by `__gnu_cxx::swap()`.

5.22.4 Member Data Documentation

5.22.4.1 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> const
 __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos
 [static]`

Value returned by various member functions when they fail.

Definition at line 77 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

5.23 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

Public Types

- `typedef _ToType::element_type * type`

5.23.1 Detailed Description

`template<typename _ToType> struct __gnu_cxx::_Caster<_ToType>`

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 45 of file `cast.h`.

The documentation for this struct was generated from the following file:

- `cast.h`

5.24 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

Mapping from character type to associated types.

Public Types

- typedef unsigned long **int_type**
- typedef [std::streamoff](#) **off_type**
- typedef [std::streampos](#) **pos_type**
- typedef std::mbstate_t **state_type**

5.24.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::_Char_types< _CharT >`

Mapping from character type to associated types.

Note

This is an implementation class for the generic version of [char_traits](#). It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, `streamoff`, `streampos`, and `mbstate_t`. Users who need a different set of types, but who don't need to change the definitions of any function defined in [char_traits](#), can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 65 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.25 `__gnu_cxx::_ExtPtr_allocator< _Tp >` Class Template Reference

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).

Public Types

- typedef [_Pointer_adapter< _Relative_pointer_impl< const _Tp > >](#) **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef [_Pointer_adapter< _Relative_pointer_impl< _Tp > >](#) **pointer**
- typedef _Tp & **reference**

- `typedef std::size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `_ExtPtr_allocator` (const [_ExtPtr_allocator](#) &__rarg) throw ()
- `template<typename _Up > _ExtPtr_allocator` (const [_ExtPtr_allocator](#)<_Up> &__rarg) throw ()
- `const std::allocator<_Tp> &_M_getUnderlyingImp` () const
- `pointer address` (reference __x) const
- `const_pointer address` (const_reference __x) const
- `pointer allocate` (size_type __n, void *__hint=0)
- `void construct` ([pointer](#) __p, const _Tp &__val)
- `template<typename... _Args> void construct` ([pointer](#) __p, _Args &&...__args)
- `void deallocate` ([pointer](#) __p, size_type __n)
- `void destroy` ([pointer](#) __p)
- `size_type max_size` () const throw ()
- `template<typename _Up > bool operator!=` (const [_ExtPtr_allocator](#)<_Up> &__rarg)
- `bool operator!=` (const [_ExtPtr_allocator](#) &__rarg)
- `template<typename _Up > bool operator==` (const [_ExtPtr_allocator](#)<_Up> &__rarg)
- `bool operator==` (const [_ExtPtr_allocator](#) &__rarg)

Friends

- `template<typename _Up > void swap` ([_ExtPtr_allocator](#)<_Up> &, [_ExtPtr_allocator](#)<_Up> &)

5.25.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_ExtPtr_allocator<_Tp>`

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).

Definition at line 52 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr_allocator.h](#)

5.26 `__gnu_cxx::_Invalid_type` Struct Reference

5.26.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of void*, const void*, and so on.

Definition at line 205 of file pointer.h.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

5.27 `__gnu_cxx::_Pointer_adapter< _Storage_policy>` Class Template Reference

Inherits `_Storage_policy`.

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef `_Storage_policy::element_type` **element_type**
- typedef [std::random_access_iterator_tag](#) **iterator_category**
- typedef [_Pointer_adapter](#) **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef [_Unqualified_type< element_type >::type](#) **value_type**

Public Member Functions

- `_Pointer_adapter` (element_type * __arg=0)
- `_Pointer_adapter` (const [_Pointer_adapter](#) & __arg)
- template<typename _Up >
 `_Pointer_adapter` (const [_Pointer_adapter](#)< _Up > & __arg)
- template<typename _Up >
 `_Pointer_adapter` (_Up * __arg)
- `operator __unspecified_bool_type` () const
- `bool operator!` () const
- `reference operator*` () const
- [_Pointer_adapter](#) & `operator++` ()
- [_Pointer_adapter](#) `operator++` (int)

- `_Pointer_adapter` & `operator+=` (unsigned short `__offset`)
- `_Pointer_adapter` & `operator+=` (int `__offset`)
- `_Pointer_adapter` & `operator+=` (unsigned int `__offset`)
- `_Pointer_adapter` & `operator+=` (long `__offset`)
- `_Pointer_adapter` & `operator+=` (unsigned long `__offset`)
- `_Pointer_adapter` & `operator+=` (short `__offset`)
- `template<typename _Up>`
`std::ptrdiff_t operator-` (const `_Pointer_adapter<_Up>` & `__rhs`) const
- `_Pointer_adapter` `operator--` (int)
- `_Pointer_adapter` & `operator--` ()
- `_Pointer_adapter` & `operator-=` (unsigned short `__offset`)
- `_Pointer_adapter` & `operator-=` (short `__offset`)
- `_Pointer_adapter` & `operator-=` (int `__offset`)
- `_Pointer_adapter` & `operator-=` (long `__offset`)
- `_Pointer_adapter` & `operator-=` (unsigned long `__offset`)
- `_Pointer_adapter` & `operator-=` (unsigned int `__offset`)
- `element_type * operator->` () const
- `_Pointer_adapter` & `operator=` (const `_Pointer_adapter` & `__arg`)
- `template<typename _Up>`
`_Pointer_adapter` & `operator=` (`_Up` * `__arg`)
- `template<typename _Up>`
`_Pointer_adapter` & `operator=` (const `_Pointer_adapter<_Up>` & `__arg`)
- `reference operator[]` (std::ptrdiff_t `__index`) const

Friends

- `_Pointer_adapter` `operator+` (const `_Pointer_adapter` & `__lhs`, short `__offset`)
- `_Pointer_adapter` `operator+` (const `_Pointer_adapter` & `__lhs`, unsigned short `__offset`)
- `_Pointer_adapter` `operator+` (unsigned long `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter` `operator+` (unsigned short `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter` `operator+` (const `_Pointer_adapter` & `__lhs`, long `__offset`)
- `_Pointer_adapter` `operator+` (long `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter` `operator+` (const `_Pointer_adapter` & `__lhs`, int `__offset`)
- `_Pointer_adapter` `operator+` (const `_Pointer_adapter` & `__lhs`, unsigned long `__offset`)
- `_Pointer_adapter` `operator+` (unsigned int `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter` `operator+` (short `__offset`, const `_Pointer_adapter` & `__rhs`)
- `_Pointer_adapter` `operator+` (int `__offset`, const `_Pointer_adapter` & `__rhs`)

- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) &__lhs, unsigned int __offset)
- `std::ptrdiff_t` **operator-** (element_type *__lhs, const [_Pointer_adapter](#) &__rhs)
- `std::ptrdiff_t` **operator-** (const [_Pointer_adapter](#) &__lhs, element_type *__rhs)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) &__lhs, unsigned short __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) &__lhs, unsigned long __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) &__lhs, int __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) &__lhs, short __offset)
- `template<typename _Up >`
`std::ptrdiff_t` **operator-** (_Up *__lhs, const [_Pointer_adapter](#) &__rhs)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) &__lhs, long __offset)
- `template<typename _Up >`
`std::ptrdiff_t` **operator-** (const [_Pointer_adapter](#) &__lhs, _Up *__rhs)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) &__lhs, unsigned int __offset)

5.27.1 Detailed Description

`template<typename _Storage_policy> class __gnu_cxx::_Pointer_adapter< _Storage_policy >`

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from [ext/cast.h](#). This allows pointer cast operations to be overloaded is necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == \_Pointer\_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == \_Pointer\_adapter<_Std_pointer_impl<const _Tp> >;
_Tp* const == const \_Pointer\_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const \_Pointer\_adapter<_Std_pointer_impl<const _Tp> >;
```

5.28 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference 855

Definition at line 281 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.28 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `_Tp * get () const`
- `bool operator< (const _Relative_pointer_impl &__rarg) const`
- `bool operator== (const _Relative_pointer_impl &__rarg) const`
- `void set (_Tp *__arg)`

5.28.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_Relative_pointer_impl<_Tp>`

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address. This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 101 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.29 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

Public Types

- `typedef const _Tp element_type`

Public Member Functions

- `const _Tp * get () const`
- `bool operator< (const _Relative_pointer_impl &__rarg) const`
- `bool operator== (const _Relative_pointer_impl &__rarg) const`
- `void set (const _Tp *__arg)`

5.29.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_Relative_pointer_impl< const _Tp >`

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 153 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.30 `__gnu_cxx::_Std_pointer_impl< _Tp >` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `_Tp * get () const`

- `bool operator< (const _Std_pointer_impl &__rarg) const`
- `bool operator== (const _Std_pointer_impl &__rarg) const`
- `void set (element_type *__arg)`

5.30.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::Std_pointer_impl< _Tp >`

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer. A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

Definition at line 58 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.31 `__gnu_cxx::Unqualified_type< _Tp >` Struct Template Reference

Public Types

- `typedef _Tp type`

5.31.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::Unqualified_type< _Tp >`

This structure accomodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

Definition at line 233 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

5.32 `__gnu_cxx::annotate_base` Struct Reference

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Inheritance diagram for `__gnu_cxx::annotate_base`:



Public Member Functions

- void **check_allocated** (size_t label)
- void **check_allocated** (void *p, size_t size)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- [std::ostream](#) & **operator<<** ([std::ostream](#) &, const [annotate_base](#) &)

5.32.1 Detailed Description

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Definition at line 94 of file `throw_allocator.h`.

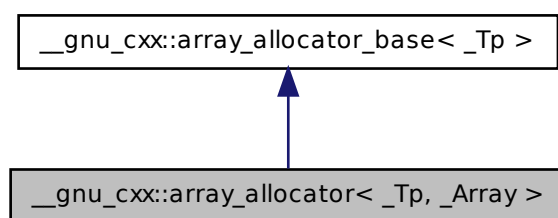
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.33 `__gnu_cxx::array_allocator< _Tp, _Array >` Class Template Reference

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Inheritance diagram for `__gnu_cxx::array_allocator< _Tp, _Array >`:



Public Types

- `typedef _Array array_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `array_allocator` (`array_type *__array=0`) `throw ()`
- `array_allocator` (`const array_allocator &__o`) `throw ()`
- `template<typename _Tp1, typename _Array1 >`
`array_allocator` (`const array_allocator< _Tp1, _Array1 > &`) `throw ()`
- `pointer address` (`reference __x`) `const`
- `const_pointer address` (`const_reference __x`) `const`
- `pointer allocate` (`size_type __n, const void *==0`)

- void **construct** (pointer __p, const _Tp &__val)
- template<typename... _Args>
void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer, size_type)
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

5.33.1 Detailed Description

template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>> class **__gnu_cxx::array_allocator**< _Tp, _Array >

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Definition at line 96 of file array_allocator.h.

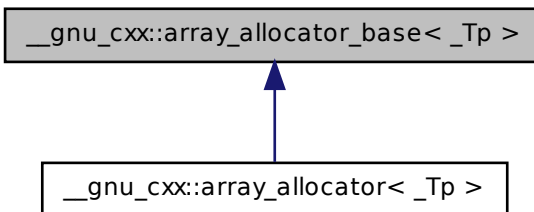
The documentation for this class was generated from the following file:

- [array_allocator.h](#)

5.34 __gnu_cxx::array_allocator_base< _Tp > Class Template Reference

Base class.

Inheritance diagram for __gnu_cxx::array_allocator_base< _Tp >:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`
- `void construct (pointer __p, const _Tp &__val)`
- `template<typename... _Args>
void construct (pointer __p, _Args &&...__args)`
- `void deallocate (pointer, size_type)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.34.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::array_allocator_base< _Tp >`

Base class.

Definition at line 46 of file `array_allocator.h`.

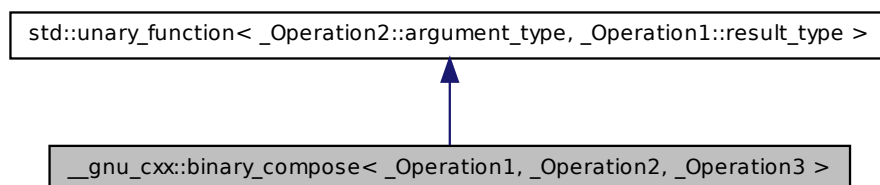
The documentation for this class was generated from the following file:

- [array_allocator.h](#)

5.35 `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >` Class Template Reference

An [SGI extension](#) .

Inheritance diagram for `__gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3>`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **binary_compose** (const `_Operation1` &__x, const `_Operation2` &__y, const `_Operation3` &__z)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &__x) const

Protected Attributes

- `_Operation1` **_M_fn1**
- `_Operation2` **_M_fn2**
- `_Operation3` **_M_fn3**

5.35.1 Detailed Description

`template<class _Operation1, class _Operation2, class _Operation3> class __gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3>`

An [SGI extension](#) .

Definition at line 149 of file `ext/functional`.

5.35.2 Member Typedef Documentation

5.35.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.35.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

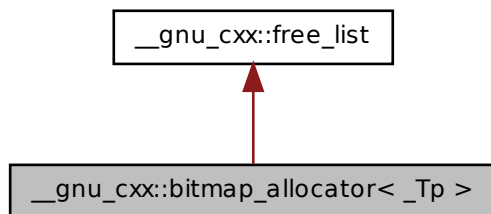
The documentation for this class was generated from the following file:

- [ext/functional](#)

5.36 `__gnu_cxx::bitmap_allocator< _Tp >` Class Template Reference

Bitmap Allocator, primary template.

Inheritance diagram for `__gnu_cxx::bitmap_allocator< _Tp >`:



Public Types

- typedef free_list::__mutex_type **__mutex_type**
- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **bitmap_allocator** (const [bitmap_allocator](#) &)
- template<typename _Tp1 >
bitmap_allocator (const [bitmap_allocator](#)< _Tp1 > &) throw ()
- pointer [_M_allocate_single_object](#) () throw (std::bad_alloc)
- void [_M_deallocate_single_object](#) (pointer __p) throw ()
- const_pointer **address** (const_reference __r) const
- pointer **address** (reference __r) const
- pointer **allocate** (size_type __n, typename [bitmap_allocator](#)< void >::const_pointer)
- pointer **allocate** (size_type __n)
- template<typename... _Args>
void **construct** (pointer __p, _Args &&...__args)
- void **construct** (pointer __p, const_reference __data)
- void **deallocate** (pointer __p, size_type __n) throw ()
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

Private Types

- typedef vector_type::iterator **iterator**
- typedef __detail::__mini_vector< value_type > **vector_type**

Private Member Functions

- void [_M_clear](#) ()
- size_t * [_M_get](#) (size_t __sz) throw (std::bad_alloc)
- void [_M_insert](#) (size_t *__addr) throw ()

5.36.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::bitmap_allocator<_Tp>`

Bitmap Allocator, primary template.

Definition at line 686 of file `bitmap_allocator.h`.

5.36.2 Member Function Documentation

5.36.2.1 `template<typename _Tp> pointer __gnu_cxx::bitmap_allocator<_Tp>::__M_allocate_single_object () throw (std::bad_alloc)`
[inline]

Allocates memory for a single object of size `sizeof(_Tp)`.

Exceptions

std::bad_alloc. If memory can not be allocated.

Complexity: Worst case complexity is $O(N)$, but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of $O(1)$! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 820 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::__Bit_scan_forward()`.

5.36.2.2 `template<typename _Tp> void __gnu_cxx::bitmap_allocator<_Tp>::__M_deallocate_single_object (pointer __p) throw ()` [inline]

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity: $O(\lg(N))$, but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in $O(1)$ time by the `deallocate` function.

Definition at line 910 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_free()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `std::__rotate()`.

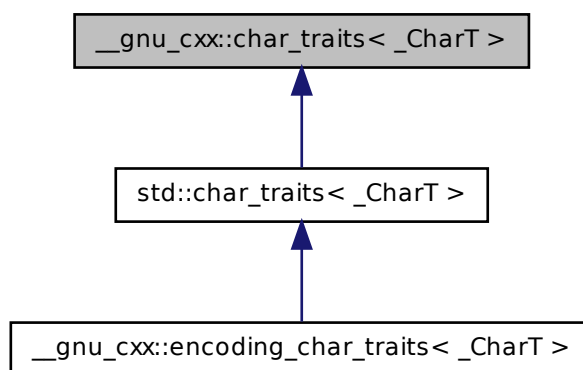
The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.37 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

Base class used to implement [std::char_traits](#).

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types<_CharT>::int_type` **int_type**
- typedef `_Char_types<_CharT>::off_type` **off_type**
- typedef `_Char_types<_CharT>::pos_type` **pos_type**
- typedef `_Char_types<_CharT>::state_type` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static `int` **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)

- static `int_type eof()`
- static `bool eq(const char_type &__c1, const char_type &__c2)`
- static `bool eq_int_type(const int_type &__c1, const int_type &__c2)`
- static `const char_type * find(const char_type *__s, std::size_t __n, const char_type &__a)`
- static `std::size_t length(const char_type *__s)`
- static `bool lt(const char_type &__c1, const char_type &__c2)`
- static `char_type * move(char_type *__s1, const char_type *__s2, std::size_t __n)`
- static `int_type not_eof(const int_type &__c)`
- static `char_type to_char_type(const int_type &__c)`
- static `int_type to_int_type(const char_type &__c)`

5.37.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::char_traits< _CharT >`

Base class used to implement [std::char_traits](#).

Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod_char_traits.h](#).

Definition at line 90 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.38 `__gnu_cxx::character< V, I, S >` Struct Template Reference

A POD class that serves as a character abstraction class.

Public Types

- typedef [character](#)< V, I, S > **char_type**
- typedef I **int_type**
- typedef S **state_type**
- typedef V **value_type**

Static Public Member Functions

- template<typename V2 >
static [char_type](#) **from** (const V2 &v)
- template<typename V2 >
static V2 **to** (const [char_type](#) &c)

Public Attributes

- value_type **value**

5.38.1 Detailed Description

template<typename V, typename I, typename S = std::mbstate_t> struct __gnu_cxx::character< V, I, S >

A POD class that serves as a character abstraction class.

Definition at line 45 of file pod_char_traits.h.

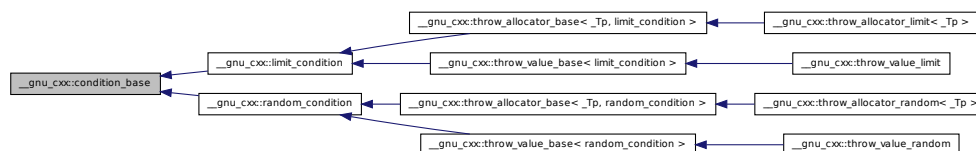
The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

5.39 __gnu_cxx::condition_base Struct Reference

Base struct for condition policy.

Inheritance diagram for `__gnu_cxx::condition_base`:



5.39.1 Detailed Description

Base struct for condition policy. Requires a public member function with the signature `void throw_conditionally()`

Definition at line 253 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.40 `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Constant_binary_fun< _Result, _Arg1, _Arg2 >`.

Public Types

- typedef `_Arg1` **first_argument_type**
- typedef `_Result` **result_type**
- typedef `_Arg2` **second_argument_type**

Public Member Functions

- **constant_binary_fun** (const `_Result` &__v)
- const `result_type` & **operator()** (const `_Arg1` &, const `_Arg2` &) const

Public Attributes

- `_Result _M_val`

5.40.1 Detailed Description

`template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1> struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`

An [SGI extension](#) .

Definition at line 315 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.41 `__gnu_cxx::constant_unary_fun< _Result, _Argument >` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Constant_unary_fun< _Result, _Argument >`.

Public Types

- `typedef _Argument argument_type`
- `typedef _Result result_type`

Public Member Functions

- `constant_unary_fun (const _Result &__v)`
- `const result_type & operator() (const _Argument &) const`

Public Attributes

- `result_type _M_val`

5.41.1 Detailed Description

```
template<class _Result, class _Argument = _Result> struct __gnu_cxx::constant_unary_fun<_Result, _Argument>
```

An [SGI extension](#).

Definition at line 307 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.42 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference

An [SGI extension](#).

Inherits `__gnu_cxx::_Constant_void_fun<_Result>`.

Public Types

- typedef `_Result` **result_type**

Public Member Functions

- **constant_void_fun** (const `_Result` &__v)
- const `result_type` & **operator()** () const

Public Attributes

- `result_type` **_M_val**

5.42.1 Detailed Description

```
template<class _Result> struct __gnu_cxx::constant_void_fun<_Result>
```

An [SGI extension](#).

Definition at line 298 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.43 `__gnu_cxx::debug_allocator< _Alloc >` Class Template Reference

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Public Types

- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef _Alloc::value_type value_type`

Public Member Functions

- `pointer allocate (size_type __n)`
- `pointer allocate (size_type __n, const void *__hint)`
- `void deallocate (pointer __p, size_type __n)`

5.43.1 Detailed Description

`template<typename _Alloc> class __gnu_cxx::debug_allocator< _Alloc >`

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 61 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug_allocator.h](#)

5.44 `__gnu_cxx::enc_filebuf<_CharT>` Class Template Reference

class `enc_filebuf`.

Inheritance diagram for `__gnu_cxx::enc_filebuf<_CharT>`:



Public Types

- typedef `codecvt<char_type, char, __state_type>` `__codecvt_type`
- typedef `__basic_file<char>` `__file_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `traits_type::state_type` `__state_type`
- typedef `basic_streambuf<char_type, traits_type>` `__streambuf_type`
- typedef `_CharT` `char_type`
- typedef `traits_type::int_type` `int_type`
- typedef `traits_type::off_type` `off_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef `traits_type::state_type` `state_type`
- typedef `encoding_char_traits<_CharT>` `traits_type`

Public Member Functions

- `enc_filebuf` (`state_type &__state`)
- `__filebuf_type * close` ()
- streamsize `in_avail` ()
- bool `is_open` () const throw ()
- `__filebuf_type * open` (const `std::string` &__s, `ios_base::openmode` __mode)
- `__filebuf_type * open` (const char *__s, `ios_base::openmode` __mode)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- streamsize `sgetn` (`char_type` *__s, streamsize __n)
- `int_type snextc` ()
- `int_type sputbackc` (`char_type` __c)
- `int_type sputc` (`char_type` __c)
- streamsize `sputn` (const `char_type` *__s, streamsize __n)
- void `stossc` ()
- `int_type sungetc` ()

Protected Member Functions

- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void **_M_create_pback** ()
- void **_M_destroy_internal_buffer** () throw ()
- void **_M_destroy_pback** () throw ()
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void **_M_set_buffer** (streamsize __off)
- bool **_M_terminate_output** ()
- void **gbump** (int __n)
- virtual void **imbue** (const locale &__loc)
- virtual int_type **overflow** (int_type __c=encoding_char_traits< _CharT >::eof())
- virtual int_type **overflow** (int_type=traits_type::eof())
- virtual int_type **pbackfail** (int_type=traits_type::eof())
- virtual int_type **pbackfail** (int_type __c=encoding_char_traits< _CharT >::eof())
- void **pbump** (int __n)
- virtual pos_type **seekoff** (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type **seekoff** (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual pos_type **seekpos** (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type **seekpos** (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual __streambuf_type * **setbuf** (char_type *__s, streamsize __n)
- virtual basic_streambuf< char_type, encoding_char_traits< _CharT > > * **setbuf** (char_type *, streamsize)
- void **setg** (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void **setp** (char_type *__pbeg, char_type *__pend)
- virtual streamsize **showmanyc** ()
- virtual int **sync** ()
- virtual int_type **uflow** ()
- virtual int_type **underflow** ()
- virtual streamsize **xsgetn** (char_type *__s, streamsize __n)
- virtual streamsize **xsgetn** (char_type *__s, streamsize __n)
- virtual streamsize **xspun** (const char_type *__s, streamsize __n)
- virtual streamsize **xspun** (const char_type *__s, streamsize __n)
- char_type * **eback** () const

- `char_type * gptr () const`
- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * epptr () const`

Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `bool _M_reading`
- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

Friends

- `__gnu_cxx::__enable_if<__is_char<_CharT2>::__value, _CharT2 * >::__-
type __copy_move_a2 (istreambuf_iterator<_CharT2>, istreambuf_iterator<
_CharT2>, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,
bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`

- `__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type` **find** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, `const _CharT2 &`)
- `basic_istream< _CharT2, _Traits2 > & getline` (`basic_istream< _CharT2, _Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`, `_CharT2`)
- class **ios_base**
- class **istreambuf_iterator**< **char_type**, **traits_type** >
- `basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2, _Traits2 > &`, `_CharT2 *`)
- `basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2, _Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`)
- class **ostreambuf_iterator**< **char_type**, **traits_type** >
- locale [pubimbue](#) (`const locale &__loc`)
- locale [getloc](#) () `const`
- `__streambuf_type * pubsetbuf` (`char_type * __s`, `streamsize __n`)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- locale `_M_buf_locale`

5.44.1 Detailed Description

`template<typename _CharT> class __gnu_cxx::enc_filebuf< _CharT >`

class [enc_filebuf](#).

Definition at line 40 of file `enc_filebuf.h`.

5.44.2 Member Typedef Documentation

5.44.2.1 `typedef basic_streambuf<char_type, traits_type> std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__streambuf_type`
[inherited]

This is a non-standard type.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

Definition at line 77 of file `fstream`.

5.44.2.2 `typedef _CharT std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

Definition at line 71 of file `fstream`.

5.44.2.3 `typedef traits_type::int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

Definition at line 73 of file `fstream`.

5.44.2.4 `typedef traits_type::off_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::off_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

Definition at line 75 of file `fstream`.

5.44.2.5 `template<typename _CharT> typedef traits_type::pos_type __gnu_cxx::enc_filebuf<_CharT>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`.

Definition at line 46 of file `enc_filebuf.h`.

5.44.2.6 `template<typename _CharT > typedef encoding_char_traits<_CharT> __gnu_cxx::enc_filebuf<_CharT>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`.

Definition at line 44 of file `enc_filebuf.h`.

5.44.3 Member Function Documentation

5.44.3.1 `void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_create_pback() [inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

5.44.3.2 `void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_destroy_pback() throw () [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file `fstream`.

5.44.3.3 `void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_set_buffer(streamsize __off) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 387 of file `fstream`.

5.44.3.4 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::close() [inherited]`

Closes the currently associated file.

Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

5.44.3.5 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eback() const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

5.44.3.6 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr() const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file `streambuf`.

5.44.3.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::eptr () const [inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `pptr()` returns the end pointer for the output sequence

Definition at line 513 of file `streambuf`.

5.44.3.8 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::gbump (int __n) [inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

5.44.3.9 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::getloc () const [inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

5.44.3.10 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr () const` [`inline`, `protected`, `inherited`]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

5.44.3.11 `virtual void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue (const locale &)` [`protected`, `virtual`, `inherited`]

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

5.44.3.15 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open (const std::string & __s, ios_base::openmode __mode) [inline, inherited]`

Opens an external file.

Parameters

- s* The name of the file.
- mode* The open mode flags.

Returns

this on success, NULL on failure

Definition at line 275 of file `fstream`.

5.44.3.16 `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::overflow (int_type = traits_type::eof()) [inline, protected, virtual, inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

- c* An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Definition at line 746 of file `streambuf`.

5.44.3.17 `virtual int_type std::basic_streambuf<_CharT,
encoding_char_traits<_CharT>>::pbackfail(int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Definition at line 702 of file `streambuf`.

5.44.3.18 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<
_CharT>>::pbase() const [inline, protected,
inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file `streambuf`.

5.44.3.19 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT
>>::pbump(int __n) [inline, protected, inherited]`

Moving the write position.

Parameters

n The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file `streambuf`.

5.44.3.20 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pptr () const` [`inline`, `protected`, `inherited`]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

5.44.3.21 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubimbue (const locale & __loc)` [`inline`, `inherited`]

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

5.44.3.22 `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out)` [`inline`, `inherited`]

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 239 of file streambuf.

5.44.3.23 `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekpos (pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 244 of file streambuf.

5.44.3.24 `__streambuf_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsetbuf (char_type * __s, streamsize __n) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file streambuf.

5.44.3.25 `int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync () [inline, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

5.44.3.26 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>::sbumpc() [inline, inherited]`

Getting the next character.

Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

5.44.3.27 `virtual pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>::seekoff(off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out) [inline, protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.44.3.28 `virtual pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>::seekpos(pos_type, ios_base::openmode = ios_base::in | ios_base::out) [inline, protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.44.3.29 `virtual __streambuf_type* std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::setbuf(char_type * __s,
streamsize __n) [protected, virtual, inherited]`

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

5.44.3.30 `virtual basic_streambuf<char_type,encoding_char_traits<_CharT
>>* std::basic_streambuf<_CharT, encoding_char_traits<
_CharT>>::setbuf(char_type *, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

5.44.3.31 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::setg (char_type * __gbeg, char_type * __gnext, char_type * __gend) [inline, protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.

gnext A pointer.

gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

5.44.3.32 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::setp (char_type * __pbeg, char_type * __pend) [inline, protected, inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

5.44.3.33 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::sgetc () [inline, inherited]`

Getting the next character.

Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

5.44.3.34 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetn(char_type * __s, streamsize __n)`
[inline, inherited]

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.44.3.35 `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::showmanyc()` **[protected, virtual, inherited]**

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
 [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

5.44.3.36 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::snextc()` **[inline, inherited]**

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

5.44.3.37 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputbackc(char_type __c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

5.44.3.38 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::putc(char_type __c) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

5.44.3.39 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputn (const char_type * __s, streamsize __n)`
[inline, inherited]

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.44.3.40 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::stoss ()` **[inline, inherited]**

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.44.3.41 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sungetc ()` **[inline, inherited]**

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

5.44.3.42 `virtual int std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::sync(void) [protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

5.44.3.43 `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::uflow() [inline, protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 678 of file `streambuf`.

5.44.3.44 `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::underflow() [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

5.44.3.45 `virtual streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::xsgetn (char_type * __s, streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

5.44.3.46 `virtual streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::xsputn (const char_type * __s, streamsize __n) [protected, virtual, inherited]`

Multiple character insertion.

Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

5.44.4 Member Data Documentation**5.44.4.1** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf` [protected, inherited]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

5.44.4.2 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

5.44.4.3 `size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_size` [protected, inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file `fstream`.

5.44.4.4 `char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf` [protected, inherited]

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 151 of file `fstream`.

5.44.4.5 `streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf_size` `[protected, inherited]`

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

5.44.4.6 `const char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_next` `[protected, inherited]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 163 of file `fstream`.

5.44.4.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_beg` `[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.44.4.8 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_cur` `[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.44.4.9 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_end` [protected, inherited]

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.44.4.10 `ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_mode` [protected, inherited]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file `fstream`.

5.44.4.11 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_beg` [protected, inherited]

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.44.4.12 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur` [protected, inherited]

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 184 of file streambuf.

5.44.4.13 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_end` [protected, inherited]

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

5.44.4.14 `char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback` [protected, inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 137 of file fstream.

5.44.4.15 `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_cur_save` [protected, inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

5.45 `__gnu_cxx::encoding_char_traits< _CharT >` Struct Template Reference 899

5.44.4.16 `char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save [protected, inherited]`

Necessary bits for putback buffer management.

Note

putbacks of over one character are not currently supported.

Definition at line 139 of file `fstream`.

5.44.4.17 `bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init [protected, inherited]`

Necessary bits for putback buffer management.

Note

putbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

5.44.4.18 `bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading [protected, inherited]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

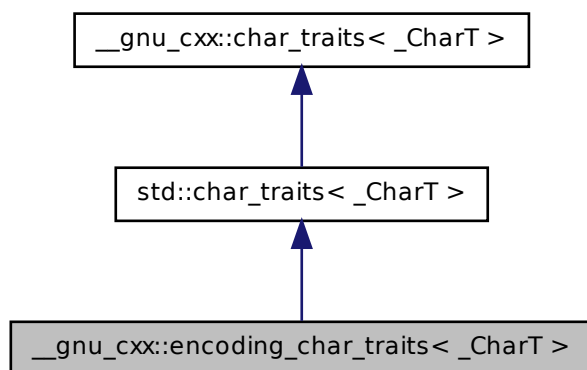
The documentation for this class was generated from the following file:

- [enc_filebuf.h](#)

5.45 `__gnu_cxx::encoding_char_traits< _CharT >` Struct Template Reference

[encoding_char_traits](#)

Inheritance diagram for `__gnu_cxx::encoding_char_traits<_CharT>`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types<_CharT>::int_type` **int_type**
- typedef `_Char_types<_CharT>::off_type` **off_type**
- typedef `std::fpos<state_type>` **pos_type**
- typedef `encoding_state` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static int_type **eof** ()
- static bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static bool **eq_int_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)

- static `std::size_t` **length** (const `char_type` *__s)
- static `bool` **lt** (const `char_type` &__c1, const `char_type` &__c2)
- static `char_type` * **move** (`char_type` *__s1, const `char_type` *__s2, `std::size_t` __n)
- static `int_type` **not_eof** (const `int_type` &__c)
- static `char_type` **to_char_type** (const `int_type` &__c)
- static `int_type` **to_int_type** (const `char_type` &__c)

5.45.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::encoding_char_traits< _CharT >`

[encoding_char_traits](#)

Definition at line 210 of file `codecvt_specializations.h`.

The documentation for this struct was generated from the following file:

- [codecvt_specializations.h](#)

5.46 `__gnu_cxx::encoding_state` Class Reference

Extension to use `iconv` for dealing with character encodings.

Public Types

- typedef `iconv_t` **descriptor_type**

Public Member Functions

- **encoding_state** (const `char` *__int, const `char` *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const [encoding_state](#) &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const `std::string` **external_encoding** () const
- bool **good** () const throw ()
- const `descriptor_type` & **in_descriptor** () const
- int **internal_bom** () const
- const `std::string` **internal_encoding** () const
- [encoding_state](#) & **operator=** (const [encoding_state](#) &__obj)
- const `descriptor_type` & **out_descriptor** () const

Protected Member Functions

- void **construct** (const [encoding_state](#) &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- [std::string](#) **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- [std::string](#) **_M_int_enc**
- descriptor_type **_M_out_desc**

5.46.1 Detailed Description

Extension to use iconv for dealing with character encodings.

Definition at line 50 of file codecvt_specializations.h.

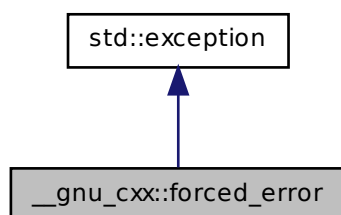
The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

5.47 `__gnu_cxx::forced_error` Struct Reference

Thrown by exception safety machinery.

Inheritance diagram for `__gnu_cxx::forced_error`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.47.1 Detailed Description

Thrown by exception safety machinery.

Definition at line 73 of file `throw_allocator.h`.

5.47.2 Member Function Documentation

5.47.2.1 virtual const char* `std::exception::what` () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and [std::ios_base::failure](#).

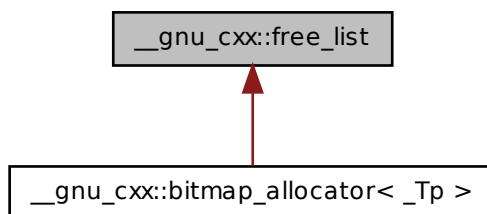
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.48 `__gnu_cxx::free_list` Class Reference

The free list class for managing chunks of memory to be given to and returned by the [bitmap_allocator](#).

Inheritance diagram for `__gnu_cxx::free_list`:



Public Types

- typedef `__mutex` **`mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t * value_type`
- typedef `__detail::__mini_vector< value_type >` **`vector_type`**

Public Member Functions

- void `_M_clear` ()
- `size_t * _M_get` (`size_t __sz`) throw (`std::bad_alloc`)
- void `_M_insert` (`size_t *__addr`) throw ()

5.48.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the [bitmap_allocator](#).

Definition at line 520 of file `bitmap_allocator.h`.

5.48.2 Member Function Documentation

5.48.2.1 `void __gnu_cxx::free_list::_M_clear ()`

This function just clears the internal Free List, and gives back all the memory to the OS.

5.48.2.2 `size_t* __gnu_cxx::free_list::_M_get (size_t __sz) throw (std::bad_alloc)`

This function gets a block of memory of the specified size from the free list.

Parameters

`__sz` The size in bytes of the memory required.

Returns

A pointer to the new memory block of size at least equal to that requested.

5.48.2.3 `void __gnu_cxx::free_list::_M_insert (size_t * __addr) throw () [inline]`

This function returns the block of memory to the internal free list.

Parameters

`__addr` The pointer to the memory block that was given by a call to the `_M_get` function.

Definition at line 630 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.49 `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc > Class` Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`

- typedef [_Ht::const_pointer](#) **const_pointer**
- typedef [_Ht::const_reference](#) **const_reference**
- typedef [_Tp](#) **data_type**
- typedef [_Ht::difference_type](#) **difference_type**
- typedef [_Ht::hasher](#) **hasher**
- typedef [_Ht::iterator](#) **iterator**
- typedef [_Ht::key_equal](#) **key_equal**
- typedef [_Ht::key_type](#) **key_type**
- typedef [_Tp](#) **mapped_type**
- typedef [_Ht::pointer](#) **pointer**
- typedef [_Ht::reference](#) **reference**
- typedef [_Ht::size_type](#) **size_type**
- typedef [_Ht::value_type](#) **value_type**

Public Member Functions

- **hash_map** (size_type __n)
- template<class [_InputIterator](#) >
hash_map ([_InputIterator](#) __f, [_InputIterator](#) __l)
- template<class [_InputIterator](#) >
hash_map ([_InputIterator](#) __f, [_InputIterator](#) __l, size_type __n)
- template<class [_InputIterator](#) >
hash_map ([_InputIterator](#) __f, [_InputIterator](#) __l, size_type __n, const hasher &__hf)
- template<class [_InputIterator](#) >
hash_map ([_InputIterator](#) __f, [_InputIterator](#) __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- **hash_map** (size_type __n, const hasher &__hf)
- **hash_map** (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- [pair](#)< iterator, iterator > **equal_range** (const key_type &__key)
- [pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)

- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_func** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- [pair](#)< iterator, bool > **insert** (const [value_type](#) &__obj)
- [pair](#)< iterator, bool > **insert_noresize** (const [value_type](#) &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- _Tp & **operator[]** (const key_type &__key)
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_map](#) &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al >
bool **operator==** (const [hash_map](#)< _K1, _T1, _HF, _EqK, _Al > &, const
[hash_map](#)< _K1, _T1, _HF, _EqK, _Al > &)

5.49.1 Detailed Description

`template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>> class __gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

This is an SGI extension.

[Todo](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 76 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash_map](#)

5.50 `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Ht::const_pointer const_pointer`
- `typedef _Ht::const_reference const_reference`
- `typedef _Tp data_type`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Tp mapped_type`
- `typedef _Ht::pointer pointer`
- `typedef _Ht::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- `hash_multimap (size_type __n)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `hash_multimap (size_type __n, const hasher &__hf)`
- `hash_multimap (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `iterator begin ()`
- `const_iterator begin () const`
- `size_type bucket_count () const`

- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- [pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- [pair](#)< iterator, iterator > **equal_range** (const key_type &__key)
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const [value_type](#) &__obj)
- iterator **insert_noresize** (const [value_type](#) &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_multimap](#) &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al >
bool **operator==** (const [hash_multimap](#)< _K1, _T1, _HF, _EqK, _Al > &, const [hash_multimap](#)< _K1, _T1, _HF, _EqK, _Al > &)

5.50.1 Detailed Description

`template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>> class __gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 289 of file hash_map.

The documentation for this class was generated from the following file:

- [hash_map](#)

5.51 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- **hash_multiset** (size_type __n)
- `template<class _InputIterator >`
hash_multiset (_InputIterator __f, _InputIterator __l)
- `template<class _InputIterator >`
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)
- `template<class _InputIterator >`
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- `template<class _InputIterator >`
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())

- `hash_multiset` (size_type __n, const hasher &__hf)
- `hash_multiset` (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- iterator `begin` () const
- size_type `bucket_count` () const
- void `clear` ()
- size_type `count` (const key_type &__key) const
- size_type `elems_in_bucket` (size_type __n) const
- bool `empty` () const
- iterator `end` () const
- `pair`< iterator, iterator > `equal_range` (const key_type &__key) const
- void `erase` (iterator __f, iterator __l)
- void `erase` (iterator __it)
- size_type `erase` (const key_type &__key)
- iterator `find` (const key_type &__key) const
- allocator_type `get_allocator` () const
- hasher `hash_funct` () const
- template<class _InputIterator >
void `insert` (_InputIterator __f, _InputIterator __l)
- iterator `insert` (const value_type &__obj)
- iterator `insert_noresize` (const value_type &__obj)
- key_equal `key_eq` () const
- size_type `max_bucket_count` () const
- size_type `max_size` () const
- void `resize` (size_type __hint)
- size_type `size` () const
- void `swap` (`hash_multiset` &hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al >
bool `operator==` (const `hash_multiset`< _Val, _HF, _EqK, _Al > &, const `hash_multiset`< _Val, _HF, _EqK, _Al > &)

5.51.1 Detailed Description

`template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>> class __gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 284 of file hash_set.

The documentation for this class was generated from the following file:

- [hash_set](#)

5.52 `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- **hash_set** (size_type __n)
- `template<class _InputIterator >`
hash_set (_InputIterator __f, _InputIterator __l)
- `template<class _InputIterator >`
hash_set (_InputIterator __f, _InputIterator __l, size_type __n)
- `template<class _InputIterator >`
hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &_hf)
- `template<class _InputIterator >`
hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &_hf, const key_equal &__eq, const allocator_type &__a=allocator_type())

- **hash_set** (size_type __n, const hasher &__hf)
- **hash_set** (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- [pair](#)< iterator, iterator > **equal_range** (const key_type &__key) const
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- [pair](#)< iterator, bool > **insert** (const value_type &__obj)
- [pair](#)< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_set](#) &__hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al >
bool **operator==** (const [hash_set](#)< _Val, _HF, _EqK, _Al > &, const [hash_set](#)< _Val, _HF, _EqK, _Al > &)

5.52.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>> class `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc>`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 83 of file hash_set.

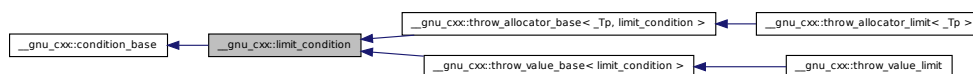
The documentation for this class was generated from the following file:

- [hash_set](#)

5.53 __gnu_cxx::limit_condition Struct Reference

Base class for incremental control and throw.

Inheritance diagram for __gnu_cxx::limit_condition:



Classes

- struct [always_adjustor](#)
Always enter the condition.
- struct [limit_adjustor](#)
*Enter the *n*th condition.*
- struct [never_adjustor](#)
Never enter the condition.

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

5.53.1 Detailed Description

Base class for incremental control and throw.

Definition at line 262 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.54 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

5.54.1 Detailed Description

Always enter the condition.

Definition at line 286 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.55 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Enter the nth condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

Public Member Functions

- `limit_adjustor` (const size_t __l)

5.55.1 Detailed Description

Enter the nth condition.

Definition at line 292 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.56 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

5.56.1 Detailed Description

Never enter the condition.

Definition at line 280 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.57 `__gnu_cxx::malloc_allocator< _Tp >` Class Template Reference

An allocator that uses `malloc`.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls `malloc`
- all deallocation calls `free`.

Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- **malloc_allocator** (const [malloc_allocator](#) &) throw ()
- template<typename _Tp1 >
 malloc_allocator (const [malloc_allocator](#)<_Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, const void *=0)
- void **construct** (pointer __p, const _Tp &__val)
- template<typename... _Args>
 void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type)
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

5.57.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::malloc_allocator<_Tp>`

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free.

Definition at line 52 of file `malloc_allocator.h`.

The documentation for this class was generated from the following file:

- [malloc_allocator.h](#)

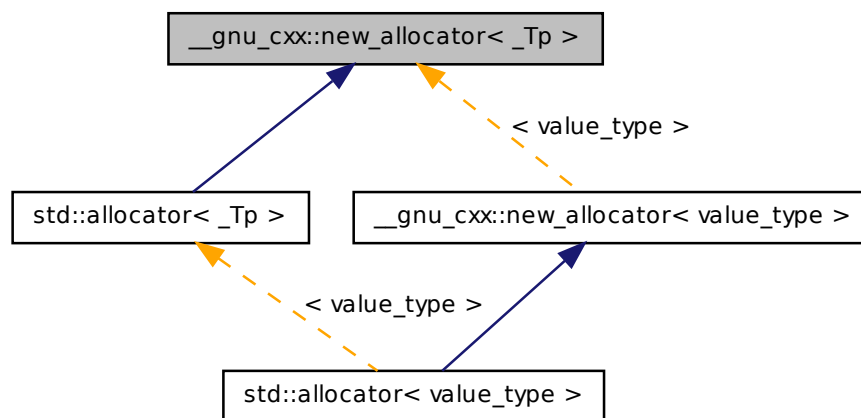
5.58 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `new_allocator (const new_allocator &) throw ()`
- `template<typename _Tp1 >
 new_allocator (const new_allocator<_Tp1 > &) throw ()`
- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`
- `pointer allocate (size_type __n, const void * = 0)`
- `void construct (pointer __p, const _Tp & __val)`

- `template<typename... _Args>`
`void construct (pointer __p, _Args &&...__args)`
- `void deallocate (pointer __p, size_type)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.58.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::new_allocator< _Tp >`

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 52 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new_allocator.h](#)

5.59 `__gnu_cxx::project1st< _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Project1st< _Arg1, _Arg2 >`.

Public Types

- `typedef _Arg1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Arg2 second_argument_type`

Public Member Functions

- `_Arg1 operator() (const _Arg1 &__x, const _Arg2 &) const`

5.59.1 Detailed Description

```
template<class _Arg1, class _Arg2> struct __gnu_cxx::project1st< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 232 of file ext/functional.

5.59.2 Member Typedef Documentation

5.59.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.59.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result
>::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.59.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.60 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Project2nd< _Arg1, _Arg2 >`.

Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

Public Member Functions

- `_Arg2 operator()` (`const _Arg1 &`, `const _Arg2 &__y`) `const`

5.60.1 Detailed Description

`template<class _Arg1, class _Arg2> struct __gnu_cxx::project2nd< _Arg1, _Arg2 >`

An [SGI extension](#) .

Definition at line 236 of file `ext/functional`.

5.60.2 Member Typedef Documentation

5.60.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`
`>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.60.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.60.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

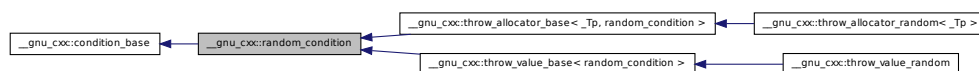
The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.61 `__gnu_cxx::random_condition` Struct Reference

Base class for random probability control and throw.

Inheritance diagram for `__gnu_cxx::random_condition`:



Classes

- struct [always_adjustor](#)
Always enter the condition.
- struct [group_adjustor](#)
Group condition.
- struct [never_adjustor](#)
Never enter the condition.

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

5.61.1 Detailed Description

Base class for random probability control and throw.

Definition at line 334 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.62 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

5.62.1 Detailed Description

Always enter the condition.

Definition at line 367 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.63 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Group condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

Public Member Functions

- `group_adjustor` (size_t size)

5.63.1 Detailed Description

Group condition.

Definition at line 352 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.64 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

5.64.1 Detailed Description

Never enter the condition.

Definition at line 361 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.65 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`.

Public Types

- `typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef const _Rb_tree_node< _Val > * _Const_Link_type`
- `typedef _Rb_tree_node< _Val > * _Link_type`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rb_tree_iterator< value_type > iterator`

- typedef `_Key` **key_type**
- typedef `value_type * pointer`
- typedef `value_type & reference`
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Val` **value_type**

Public Member Functions

- **rb_tree** (const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- bool **__rb_verify** () const
- const `_Node_allocator` & **_M_get_Node_allocator** () const
- `_Node_allocator` & **_M_get_Node_allocator** ()
- iterator **_M_insert_equal** (const `value_type` &__x)
- template<typename `_InputIterator` >
void **_M_insert_equal** (`_InputIterator` __first, `_InputIterator` __last)
- template<class `_II` >
void **_M_insert_equal** (`_II` __first, `_II` __last)
- iterator **_M_insert_equal_** (const_iterator __position, const `value_type` &__x)
- `pair`< iterator, bool > **_M_insert_unique** (const `value_type` &__x)
- template<class `_II` >
void **_M_insert_unique** (`_II` __first, `_II` __last)
- template<typename `_InputIterator` >
void **_M_insert_unique** (`_InputIterator` __first, `_InputIterator` __last)
- iterator **_M_insert_unique_** (const_iterator __position, const `value_type` &__x)
- const_iterator **begin** () const
- iterator **begin** ()
- void **clear** ()
- `size_type` **count** (const `key_type` &__k) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- `pair`< const_iterator, const_iterator > **equal_range** (const `key_type` &__k) const
- `pair`< iterator, iterator > **equal_range** (const `key_type` &__k)
- void **erase** (const `key_type` *__first, const `key_type` *__last)
- const_iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (iterator __position)
- const_iterator **erase** (const_iterator __position)
- `size_type` **erase** (const `key_type` &__x)
- iterator **erase** (iterator __first, iterator __last)

- iterator **find** (const key_type &__k)
- const_iterator **find** (const key_type &__k) const
- allocator_type **get_allocator** () const
- _Compare **key_comp** () const
- const_iterator **lower_bound** (const key_type &__k) const
- iterator **lower_bound** (const key_type &__k)
- size_type **max_size** () const
- [reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rend** () const
- [reverse_iterator](#) **rend** ()
- size_type **size** () const
- void **swap** (_Rb_tree &__t)
- iterator **upper_bound** (const key_type &__k)
- const_iterator **upper_bound** (const key_type &__k) const

Protected Types

- typedef _Rb_tree_node_base * **_Base_ptr**
- typedef const _Rb_tree_node_base * **_Const_Base_ptr**

Protected Member Functions

- _Link_type **_M_begin** ()
- _Const_Link_type **_M_begin** () const
- _Link_type **_M_clone_node** (_Const_Link_type __x)
- template<typename... _Args>
_Link_type **_M_create_node** (_Args &&...__args)
- void **_M_destroy_node** (_Link_type __p)
- _Const_Link_type **_M_end** () const
- _Link_type **_M_end** ()
- _Link_type **_M_get_node** ()
- _Base_ptr & **_M_leftmost** ()
- _Const_Base_ptr **_M_leftmost** () const
- void **_M_put_node** (_Link_type __p)
- _Base_ptr & **_M_rightmost** ()
- _Const_Base_ptr **_M_rightmost** () const
- _Base_ptr & **_M_root** ()
- _Const_Base_ptr **_M_root** () const

Static Protected Member Functions

- static const `_Key` & `_S_key` (`_Const_Link_type __x`)
- static const `_Key` & `_S_key` (`_Const_Base_ptr __x`)
- static `_Link_type` `_S_left` (`_Base_ptr __x`)
- static `_Const_Link_type` `_S_left` (`_Const_Base_ptr __x`)
- static `_Base_ptr` `_S_maximum` (`_Base_ptr __x`)
- static `_Const_Base_ptr` `_S_maximum` (`_Const_Base_ptr __x`)
- static `_Base_ptr` `_S_minimum` (`_Base_ptr __x`)
- static `_Const_Base_ptr` `_S_minimum` (`_Const_Base_ptr __x`)
- static `_Const_Link_type` `_S_right` (`_Const_Base_ptr __x`)
- static `_Link_type` `_S_right` (`_Base_ptr __x`)
- static const_reference `_S_value` (`_Const_Link_type __x`)
- static const_reference `_S_value` (`_Const_Base_ptr __x`)

Protected Attributes

- `_Rb_tree_impl<_Compare>` `_M_impl`

5.65.1 Detailed Description

`template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = allocator<_Value>> struct __gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc>`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 78 of file `rb_tree`.

The documentation for this struct was generated from the following file:

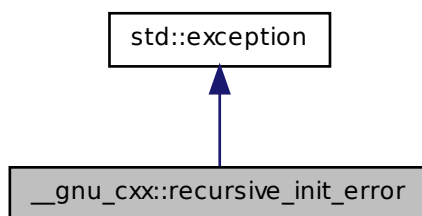
- [rb_tree](#)

5.66 `__gnu_cxx::recursive_init_error` Class Reference

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Inheritance diagram for `__gnu_cxx::recursive_init_error`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.66.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined. Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 620 of file `cxxabi.h`.

5.66.2 Member Function Documentation

5.66.2.1 virtual const char* std::exception::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and

[std::ios_base::failure](#).

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

5.67 `__gnu_cxx::rope< _CharT, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::Rope_base< _CharT, _Alloc >`.

Public Types

- `typedef _Rope_RopeConcatenation< _CharT, _Alloc > __C`
- `typedef _Rope_RopeFunction< _CharT, _Alloc > __F`
- `typedef _Rope_RopeLeaf< _CharT, _Alloc > __L`
- `typedef _Rope_RopeSubstring< _CharT, _Alloc > __S`
- `typedef _Alloc::template rebind< __C >::other _CAlloc`
- `typedef _Alloc::template rebind< _CharT >::other _DataAlloc`
- `typedef _Alloc::template rebind< __F >::other _FAlloc`
- `typedef _Alloc::template rebind< __L >::other _LAlloc`
- `typedef _Alloc::template rebind< __S >::other _SAlloc`
- `typedef _Rope_const_iterator< _CharT, _Alloc > const_iterator`
- `typedef const _CharT * const_pointer`
- `typedef _CharT const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rope_iterator< _CharT, _Alloc > iterator`
- `typedef _Rope_char_ptr_proxy< _CharT, _Alloc > pointer`
- `typedef _Rope_char_ref_proxy< _CharT, _Alloc > reference`
- `typedef std::reverse_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _CharT value_type`

Public Member Functions

- **rope** (const _CharT * __s, const allocator_type & __a=allocator_type())
- **rope** (const _CharT * __s, size_t __len, const allocator_type & __a=allocator_type())
- **rope** (const iterator & __s, const iterator & __e, const allocator_type & __a=allocator_type())

- **rope** (_CharT __c, const allocator_type &__a=allocator_type())
- **rope** (size_t __n, _CharT __c, const allocator_type &__a=allocator_type())
- **rope** (const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, const _CharT *__e, const allocator_type &__a=allocator_type())
- **rope** (char_producer< _CharT > *__fn, size_t __len, bool __delete_fn, const allocator_type &__a=allocator_type())
- **rope** (const rope &__x, const allocator_type &__a=allocator_type())
- **rope** (const const_iterator &__s, const const_iterator &__e, const allocator_type &__a=allocator_type())
- allocator_type & **M_get_allocator** ()
- const allocator_type & **M_get_allocator** () const
- rope & **append** (const _CharT *__iter, size_t __n)
- rope & **append** (const _CharT *__c_string)
- rope & **append** (const _CharT *__s, const _CharT *__e)
- rope & **append** (const_iterator __s, const_iterator __e)
- rope & **append** (_CharT __c)
- rope & **append** ()
- rope & **append** (const rope &__y)
- rope & **append** (size_t __n, _CharT __c)
- void **apply_to_pieces** (size_t __begin, size_t __end, _Rope_char_consumer< _CharT > &__c) const
- _CharT **at** (size_type __pos) const
- _CharT **back** () const
- void **balance** ()
- const_iterator **begin** () const
- const_iterator **begin** ()
- const _CharT * **c_str** () const
- void **clear** ()
- int **compare** (const rope &__y) const
- const_iterator **const_begin** () const
- const_iterator **const_end** () const
- const_reverse_iterator **const_rbegin** () const
- const_reverse_iterator **const_rend** () const
- void **copy** (_CharT *__buffer) const
- size_type **copy** (size_type __pos, size_type __n, _CharT *__buffer) const
- void **delete_c_str** ()
- void **dump** ()
- bool **empty** () const
- const_iterator **end** () const
- const_iterator **end** ()
- void **erase** (size_t __p, size_t __n)
- void **erase** (size_t __p)

- iterator **erase** (const iterator &__p, const iterator &__q)
- iterator **erase** (const iterator &__p)
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find** (const _CharT *__s, size_type __pos=0) const
- _CharT **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- iterator **insert** (const iterator &__p, const [rope](#) &__r)
- iterator **insert** (const iterator &__p, size_t __n, _CharT __c)
- iterator **insert** (const iterator &__p, _CharT __c)
- iterator **insert** (const iterator &__p)
- iterator **insert** (const iterator &__p, const _CharT *c_string)
- iterator **insert** (const iterator &__p, const _CharT *__i, size_t __n)
- iterator **insert** (const iterator &__p, const const_iterator &__i, const const_iterator &__j)
- iterator **insert** (const iterator &__p, const iterator &__i, const iterator &__j)
- void **insert** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **insert** (size_t __p, size_t __n, _CharT __c)
- void **insert** (size_t __p, const _CharT *__i, size_t __n)
- void **insert** (size_t __p)
- void **insert** (size_t __p, const [rope](#) &__r)
- void **insert** (size_t __p, _CharT __c)
- void **insert** (size_t __p, const _CharT *c_string)
- void **insert** (size_t __p, const _CharT *__i, const _CharT *__j)
- void **insert** (size_t __p, const iterator &__i, const iterator &__j)
- size_type **length** () const
- size_type **max_size** () const
- iterator **mutable_begin** ()
- iterator **mutable_end** ()
- [reverse_iterator](#) **mutable_rbegin** ()
- reference **mutable_reference_at** (size_type __pos)
- [reverse_iterator](#) **mutable_rend** ()
- [rope](#) & **operator=** (const [rope](#) &__x)
- _CharT **operator[]** (size_type __pos) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_CharT __x)
- void **push_front** (_CharT __x)
- [const_reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- void **replace** (size_t __p, const iterator &__i, const iterator &__j)

- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, const _CharT *__j)
- void **replace** (size_t __p, const _CharT *__i, size_t __i_len)
- void **replace** (const iterator &__p, const iterator &__q, _CharT __c)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__c_string)
- void **replace** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, iterator __i, iterator __j)
- void **replace** (const iterator &__p, const rope &__r)
- void **replace** (const iterator &__p, const iterator &__q, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, size_t __n, const rope &__r)
- void **replace** (size_t __p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const rope &__r)
- void **replace** (const iterator &__p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__i, size_t __n)
- void **replace** (size_t __p, size_t __n, const _CharT *__c_string)
- void **replace** (size_t __p, size_t __n, _CharT __c)
- void **replace** (size_t __p, size_t __n, const _CharT *__i, size_t __i_len)
- void **replace** (size_t __p, const rope &__r)
- void **replace** (size_t __p, const _CharT *__c_string)
- void **replace** (size_t __p, size_t __n, const iterator &__i, const iterator &__j)
- void **replace** (size_t __p, size_t __n, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__c_string)
- void **replace** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, size_t __n, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const iterator &__i, const iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, size_t __n)
- const _CharT * **replace_with_c_str** ()
- size_type **size** () const
- rope< _CharT, _Alloc > **substr** (const_iterator __start)
- rope **substr** (const_iterator __start, const_iterator __end) const
- rope **substr** (iterator __start) const
- rope **substr** (size_t __start, size_t __len=1) const
- rope **substr** (iterator __start, iterator __end) const
- void **swap** (rope &__b)

Static Public Member Functions

- static `__C * _C_allocate` (size_t __n)
- static void `_C_deallocate` (__C *__p, size_t __n)
- static `_CharT * _Data_allocate` (size_t __n)
- static void `_Data_deallocate` (_CharT *__p, size_t __n)
- static `__F * _F_allocate` (size_t __n)
- static void `_F_deallocate` (__F *__p, size_t __n)
- static `__L * _L_allocate` (size_t __n)
- static void `_L_deallocate` (__L *__p, size_t __n)
- static `__S * _S_allocate` (size_t __n)
- static void `_S_deallocate` (__S *__p, size_t __n)

Public Attributes

- `_RopeRep * _M_tree_ptr`

Static Public Attributes

- static const size_type `npos`

Protected Types

- enum { `_S_copy_max` }
- typedef `_Rope_base<_CharT, _Alloc> _Base`
- typedef `_CharT * _Cstrptr`
- typedef `_Rope_RopeConcatenation<_CharT, _Alloc> _RopeConcatenation`
- typedef `_Rope_RopeFunction<_CharT, _Alloc> _RopeFunction`
- typedef `_Rope_RopeLeaf<_CharT, _Alloc> _RopeLeaf`
- typedef `_Rope_RopeRep<_CharT, _Alloc> _RopeRep`
- typedef `_Rope_RopeSubstring<_CharT, _Alloc> _RopeSubstring`
- typedef `_Rope_self_destruct_ptr<_CharT, _Alloc> _Self_destruct_ptr`
- typedef `_Base::allocator_type allocator_type`

Static Protected Member Functions

- static size_t `_S_allocated_capacity` (size_t __n)
- static bool `_S_apply_to_pieces` (_Rope_char_consumer<_CharT> &__c, const _RopeRep *__r, size_t __begin, size_t __end)
- static `_RopeRep * _S_concat` (_RopeRep *__left, _RopeRep *__right)

- static `_RopeRep * _S_concat_char_iter` (`_RopeRep * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_RopeRep * _S_destr_concat_char_iter` (`_RopeRep * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_RopeLeaf * _S_destr_leaf_concat_char_iter` (`_RopeLeaf * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_CharT _S_fetch` (`_RopeRep * __r`, `size_type __pos`)
- static `_CharT * _S_fetch_ptr` (`_RopeRep * __r`, `size_type __pos`)
- static `bool _S_is0` (`_CharT __c`)
- static `_RopeLeaf * _S_leaf_concat_char_iter` (`_RopeLeaf * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_RopeConcatenation * _S_new_RopeConcatenation` (`_RopeRep * __left`, `_RopeRep * __right`, `allocator_type & __a`)
- static `_RopeFunction * _S_new_RopeFunction` (`char_producer< _CharT > * __f`, `size_t __size`, `bool __d`, `allocator_type & __a`)
- static `_RopeLeaf * _S_new_RopeLeaf` (`_CharT * __s`, `size_t __size`, `allocator_type & __a`)
- static `_RopeSubstring * _S_new_RopeSubstring` (`_Rope_RopeRep< _CharT, _Alloc > * __b`, `size_t __s`, `size_t __l`, `allocator_type & __a`)
- static `void _S_ref` (`_RopeRep * __t`)
- static `_RopeLeaf * _S_RopeLeaf_from_unowned_char_ptr` (const `_CharT * __s`, `size_t __size`, `allocator_type & __a`)
- static `size_t _S_rounded_up_size` (`size_t __n`)
- static `_RopeRep * _S_substring` (`_RopeRep * __base`, `size_t __start`, `size_t __endp1`)
- static `_RopeRep * _S_tree_concat` (`_RopeRep * __left`, `_RopeRep * __right`)
- static `void _S_unref` (`_RopeRep * __t`)
- static `_RopeRep * replace` (`_RopeRep * __old`, `size_t __pos1`, `size_t __pos2`, `_RopeRep * __r`)

Static Protected Attributes

- static `_CharT _S_empty_c_str` [1]

Friends

- class `_Rope_char_ptr_proxy< _CharT, _Alloc >`
- class `_Rope_char_ref_proxy< _CharT, _Alloc >`
- class `_Rope_const_iterator< _CharT, _Alloc >`
- class `_Rope_iterator< _CharT, _Alloc >`
- class `_Rope_iterator_base< _CharT, _Alloc >`
- struct `_Rope_RopeRep< _CharT, _Alloc >`

- struct `_Rope_RopeSubstring< _CharT, _Alloc >`
- template<class `_CharT2`, class `_Alloc2` >
`rope< _CharT2, _Alloc2 > operator+` (const `rope< _CharT2, _Alloc2 >` &__left, const `rope< _CharT2, _Alloc2 >` &__right)
- template<class `_CharT2`, class `_Alloc2` >
`rope< _CharT2, _Alloc2 > operator+` (const `rope< _CharT2, _Alloc2 >` &__left, `_CharT2` __right)
- template<class `_CharT2`, class `_Alloc2` >
`rope< _CharT2, _Alloc2 > operator+` (const `rope< _CharT2, _Alloc2 >` &__left, const `_CharT2 *`__right)

5.67.1 Detailed Description

`template<class _CharT, class _Alloc> class __gnu_cxx::rope< _CharT, _Alloc >`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 1508 of file rope.

The documentation for this class was generated from the following files:

- `rope`
- `ropeimpl.h`

5.68 `__gnu_cxx::select1st< _Pair >` Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select1st< _Pair >`.

Public Types

- typedef `_Pair` `argument_type`
- typedef `_Pair::first_type` `result_type`

Public Member Functions

- `_Pair::first_type & operator() (_Pair &__x) const`
- `const _Pair::first_type & operator() (const _Pair &__x) const`

5.68.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select1st< _Pair >`

An [SGI extension](#) .

Definition at line 198 of file `ext/functional`.

5.68.2 Member Typedef Documentation

5.68.2.1 `typedef _Pair std::unary_function< _Pair , _Pair::first_type >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.68.2.2 `typedef _Pair::first_type std::unary_function< _Pair , _Pair::first_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.69 __gnu_cxx::select2nd< _Pair > Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select2nd< _Pair >`.

Public Types

- `typedef _Pair argument_type`

- `typedef _Pair::second_type` [result_type](#)

Public Member Functions

- `_Pair::second_type & operator() (_Pair &__x) const`
- `const _Pair::second_type & operator() (const _Pair &__x) const`

5.69.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select2nd< _Pair >`

An [SGI extension](#) .

Definition at line 202 of file `ext/functional`.

5.69.2 Member Typedef Documentation

5.69.2.1 `typedef _Pair std::unary_function< _Pair , _Pair::second_type >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.69.2.2 `typedef _Pair::second_type std::unary_function< _Pair , _Pair::second_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.70 `__gnu_cxx::slist< _Tp, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::Slist_base< _Tp, _Alloc >`.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Slist_iterator< _Tp, const _Tp &, const _Tp * > **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Slist_iterator< _Tp, _Tp &, _Tp * > **iterator**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **slist** (const allocator_type &__a=allocator_type())
- **slist** (size_type __n)
- template<class _InputIterator >
 slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- **slist** (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())
- **slist** (const [slist](#) &__x)
- template<class _Integer >
 void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<class _InputIterator >
 void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_fill_assign** (size_type __n, const _Tp &__val)
- void **assign** (size_type __n, const _Tp &__val)
- template<class _InputIterator >
 void **assign** (_InputIterator __first, _InputIterator __last)
- iterator **before_begin** ()
- const_iterator **before_begin** () const
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __pos)
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase_after** (iterator __pos)

- iterator **erase_after** (iterator __before_first, iterator __last)
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (iterator __pos, const value_type &__x)
- iterator **insert** (iterator __pos)
- void **insert** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
void **insert** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert_after** (iterator __pos)
- void **insert_after** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
void **insert_after** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert_after** (iterator __pos, const value_type &__x)
- size_type **max_size** () const
- void **merge** (slist &__x)
- template<class _StrictWeakOrdering >
void **merge** (slist &, _StrictWeakOrdering)
- slist & **operator=** (const slist &__x)
- void **pop_front** ()
- iterator **previous** (const_iterator __pos)
- const_iterator **previous** (const_iterator __pos) const
- void **push_front** ()
- void **push_front** (const value_type &__x)
- void **remove** (const _Tp &__val)
- template<class _Predicate >
void **remove_if** (_Predicate __pred)
- void **resize** (size_type new_size, const _Tp &__x)
- void **resize** (size_type new_size)
- void **reverse** ()
- size_type **size** () const
- void **sort** ()
- template<class _StrictWeakOrdering >
void **sort** (_StrictWeakOrdering __comp)
- void **splice** (iterator __pos, slist &__x, iterator __i)
- void **splice** (iterator __pos, slist &__x)
- void **splice** (iterator __pos, slist &__x, iterator __first, iterator __last)
- void **splice_after** (iterator __pos, slist &__x)
- void **splice_after** (iterator __pos, iterator __prev)
- void **splice_after** (iterator __pos, iterator __before_first, iterator __before_last)
- void **swap** (slist &__x)
- template<class _BinaryPredicate >
void **unique** (_BinaryPredicate __pred)
- void **unique** ()

Private Types

- `typedef _Alloc::template rebind< _Slist_node< _Tp > >::other _Node_alloc`

Private Member Functions

- `_Slist_node_base * _M_erase_after (_Slist_node_base *__pos)`
- `_Slist_node_base * _M_erase_after (_Slist_node_base *, _Slist_node_base *)`
- `_Slist_node< _Tp > * _M_get_node ()`
- `void _M_put_node (_Slist_node< _Tp > *__p)`

Private Attributes

- `_Slist_node_base _M_head`

5.70.1 Detailed Description

`template<class _Tp, class _Alloc = allocator<_Tp>> class __gnu_cxx::slist< _Tp, _Alloc >`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 291 of file `slist`.

The documentation for this class was generated from the following file:

- [slist](#)

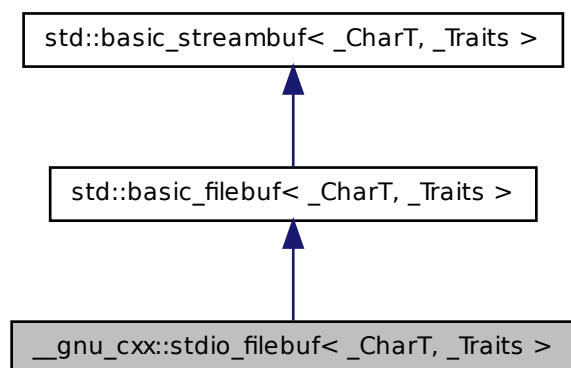
5.71 `__gnu_cxx::stdio_filebuf< _CharT, _Traits >` Class Template Reference

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 941

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



Public Types

- typedef `codecvt<char_type, char, __state_type>` `__codecvt_type`
- typedef `__basic_file<char>` `__file_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `traits_type::state_type` `__state_type`
- typedef `basic_streambuf<char_type, traits_type>` `__streambuf_type`
- typedef `_CharT` `char_type`
- typedef `traits_type::int_type` `int_type`
- typedef `traits_type::off_type` `off_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef `std::size_t` `size_t`
- typedef `_Traits` `traits_type`

Public Member Functions

- `stdio_filebuf()`
- `stdio_filebuf(int __fd, std::ios_base::openmode __mode, size_t __size=static_cast<size_t>(BUFSIZ))`
- `stdio_filebuf(std::__c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast<size_t>(BUFSIZ))`

- virtual `~stdio_filebuf` ()
- `__filebuf_type` * `close` ()
- `int` `fd` ()
- `std::__c_file` * `file` ()
- `streamsize` `in_avail` ()
- `bool` `is_open` () const throw ()
- `__filebuf_type` * `open` (const `char` *`__s`, `ios_base::openmode` `__mode`)
- `__filebuf_type` * `open` (const `std::string` &`__s`, `ios_base::openmode` `__mode`)
- `int_type` `sbumpc` ()
- `int_type` `sgetc` ()
- `streamsize` `sgetn` (`char_type` *`__s`, `streamsize` `__n`)
- `int_type` `snextc` ()
- `int_type` `sputbackc` (`char_type` `__c`)
- `int_type` `sputc` (`char_type` `__c`)
- `streamsize` `sputn` (const `char_type` *`__s`, `streamsize` `__n`)
- void `stoss` ()
- `int_type` `sungetc` ()

Protected Member Functions

- void `_M_allocate_internal_buffer` ()
- `bool` `_M_convert_to_external` (`char_type` *, `streamsize`)
- void `_M_create_pback` ()
- void `_M_destroy_internal_buffer` () throw ()
- void `_M_destroy_pback` () throw ()
- `pos_type` `_M_seek` (`off_type` `__off`, `ios_base::seekdir` `__way`, `__state_type` `__state`)
- void `_M_set_buffer` (`streamsize` `__off`)
- `bool` `_M_terminate_output` ()
- void `gbump` (`int` `__n`)
- virtual void `imbue` (const `locale` &`__loc`)
- virtual `int_type` `overflow` (`int_type` `__c`=`_Traits::eof()`)
- virtual `int_type` `overflow` (`int_type`=`traits_type::eof()`)
- virtual `int_type` `pbackfail` (`int_type`=`traits_type::eof()`)
- virtual `int_type` `pbackfail` (`int_type` `__c`=`_Traits::eof()`)
- void `pbump` (`int` `__n`)
- virtual `pos_type` `seekoff` (`off_type` `__off`, `ios_base::seekdir` `__way`, `ios_base::openmode` `__mode`=`ios_base::in|ios_base::out`)
- virtual `pos_type` `seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode`=`ios_base::in|ios_base::out`)
- virtual `pos_type` `seekpos` (`pos_type` `__pos`, `ios_base::openmode` `__mode`=`ios_base::in|ios_base::out`)

- virtual `pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)`
- virtual `__streambuf_type * setbuf (char_type *__s, streamsize __n)`
- virtual `basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)`
- void `setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)`
- void `setp (char_type *__pbeg, char_type *__pend)`
- virtual `streamsize showmanyc ()`
- virtual `int sync ()`
- virtual `int_type uflow ()`
- virtual `int_type underflow ()`
- virtual `streamsize xsgetn (char_type *__s, streamsize __n)`
- virtual `streamsize xsgetn (char_type *__s, streamsize __n)`
- virtual `streamsize xspun (const char_type *__s, streamsize __n)`
- virtual `streamsize xspun (const char_type *__s, streamsize __n)`
- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * ep_ptr () const`

Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `bool _M_reading`
- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
 - `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`
`bool &)`
 - `class basic_ios< char_type, traits_type >`
 - `class basic_istream< char_type, traits_type >`
 - `class basic_ostream< char_type, traits_type >`
 - `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf-`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _-`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
 - `class ios_base`
 - `class istreambuf_iterator< char_type, traits_type >`
 - `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
 - `class ostreambuf_iterator< char_type, traits_type >`
-
- `locale pubimbue (const locale &__loc)`
 - `locale getloc () const`
 - `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
 - `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_-`
`base::openmode __mode=ios_base::in|ios_base::out)`
 - `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_-`
`base::in|ios_base::out)`
 - `int pubsync ()`
 - `char_type * _M_in_beg`
 - `char_type * _M_in_cur`
 - `char_type * _M_in_end`
 - `char_type * _M_out_beg`
 - `char_type * _M_out_cur`
 - `char_type * _M_out_end`
 - `locale _M_buf_locale`

5.71.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 49 of file `stdio_filebuf.h`.

5.71.2 Member Typedef Documentation

5.71.2.1 `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_filebuf<_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 77 of file `fstream`.

5.71.2.2 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_filebuf<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf<_CharT, _Traits>](#).

Definition at line 53 of file `stdio_filebuf.h`.

5.71.2.3 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_filebuf<_CharT, _Traits>::__int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf<_CharT, _Traits>](#).

Definition at line 55 of file `stdio_filebuf.h`.

5.71.2.4 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> typedef traits_type::off_type
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf< _CharT, _Traits >](#).

Definition at line 57 of file `stdio_filebuf.h`.

5.71.2.5 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> typedef traits_type::pos_type
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf< _CharT, _Traits >](#).

Definition at line 56 of file `stdio_filebuf.h`.

5.71.2.6 `template<typename _CharT, typename _Traits = std::char_traits<_
_CharT>> typedef _Traits __gnu_cxx::stdio_filebuf< _CharT, _Traits
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf< _CharT, _Traits >](#).

Definition at line 54 of file `stdio_filebuf.h`.

5.71.3 Constructor & Destructor Documentation

5.71.3.1 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf< _CharT,
_Traits >::stdio_filebuf () [inline]`

deferred initialization

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 947

Definition at line 64 of file `stdio_filebuf.h`.

```
5.71.3.2 template<typename _CharT, typename _Traits >
    __gnu_cxx::stdio_filebuf<_CharT, _Traits >::stdio_filebuf
    ( int __fd, std::ios_base::openmode __mode, size_t __size =
      static_cast<size_t>(BUFSIZ) )
```

Parameters

fd An open file descriptor.

mode Same meaning as in a standard filebuf.

size Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 126 of file `stdio_filebuf.h`.

References `std::basic_filebuf<_CharT, _Traits >::M_buf_size`, `std::basic_filebuf<_CharT, _Traits >::M_mode`, `std::basic_filebuf<_CharT, _Traits >::M_reading`, `std::basic_filebuf<_CharT, _Traits >::M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits >::is_open()`.

```
5.71.3.3 template<typename _CharT, typename _Traits >
    __gnu_cxx::stdio_filebuf<_CharT, _Traits >::stdio_filebuf (
    std::_c_file * __f, std::ios_base::openmode __mode, size_t __size =
    static_cast<size_t>(BUFSIZ) )
```

Parameters

f An open `FILE*`.

mode Same meaning as in a standard filebuf.

size Optimal or preferred size of internal buffer, in chars. Defaults to system's `BUFSIZ`.

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 142 of file `stdio_filebuf.h`.

References `std::basic_filebuf<_CharT, _Traits >::M_buf_size`, `std::basic_filebuf<_CharT, _Traits >::M_mode`, `std::basic_filebuf<_CharT, _Traits >::M_reading`, `std::basic_filebuf<_CharT, _Traits >::M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits >::is_open()`.

5.71.3.4 `template<typename _CharT, typename _Traits >
 __gnu_cxx::stdio_filebuf< _CharT, _Traits >::~~stdio_filebuf ()
 [virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 121 of file `stdio_filebuf.h`.

5.71.4 Member Function Documentation

5.71.4.1 `template<typename _CharT, typename _Traits> void
 std::basic_filebuf< _CharT, _Traits >::_M_create_pback ()
 [inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

5.71.4.2 `template<typename _CharT, typename _Traits> void
 std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw
 () [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.4.3 `template<typename _CharT, typename _Traits> void
 std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (streamsize
 __off) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 387 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 949

5.71.4.4 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT, _Traits>::close () [inherited]`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

5.71.4.5 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback () const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.4.6 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr () const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.4.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [eptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xspn()`.

5.71.4.8 `template<typename _CharT , typename _Traits =
std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf< _CharT,
_Traits >::fd () [inline]`

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 107 of file `stdio_filebuf.h`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 951

5.71.4.9 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> std::__c_file* __gnu_cxx::stdio_filebuf<
_CharT, _Traits>::file () [inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 117 of file `stdio_filebuf.h`.

5.71.4.10 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

5.71.4.11 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

5.71.4.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::gptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.4.13 `template<typename _CharT, typename _Traits > void
std::basic_filebuf< _CharT, _Traits >::imbue (const locale &)
[protected, virtual, inherited]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 855 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 953

5.71.4.14 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::in_avail () [inline,
inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file streambuf.

5.71.4.15 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf<_CharT, _Traits>::is_open () const throw ()
[inline, inherited]`

Returns true if the external file is open.

Definition at line 222 of file fstream.

Referenced by `std::basic_filebuf<_CharT, _Traits>::close()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, and `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`.

5.71.4.16 `template<typename _CharT, typename _Traits> basic_filebuf<
_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT,
_Traits>::open (const char * __s, ios_base::openmode __mode)
[inherited]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Returns

this on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +-----

	ios_base	Flag combination	stdio equivalent
app			in out trunc
in out app			
in app			
binary app			
binary in out app			
binary in app			

Definition at line 94 of file fstream.tcc.

References std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), std::ios_base::ate, std::basic_filebuf< _CharT, _Traits >::close(), std::ios_base::end, and std::basic_filebuf< _CharT, _Traits >::is_open().

5.71.4.17 `template<typename _CharT, typename _Traits> __filebuf_type*
std::basic_filebuf< _CharT, _Traits >::open (const std::string &
__s, ios_base::openmode __mode) [inline, inherited]`

Opens an external file.

Parameters

- s* The name of the file.
- mode* The open mode flags.

Returns

this on success, NULL on failure

Definition at line 275 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::open().

5.71.4.18 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::overflow (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

- c* An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually `c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 746 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.71.4.19 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail(int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

`c` The character to be inserted back into the sequence.

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 702 of file `streambuf`.

5.71.4.20 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

5.71.4.21 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.71.4.22 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence

5.71 `__gnu_cxx::stdio_filebuf< _CharT, _Traits >` Class Template Reference 957

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.71.4.23 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

5.71.4.24 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

5.71.4.25 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.71.4.26 `template<typename _CharT, typename _Traits> __streambuf_type*
std::basic_streambuf< _CharT, _Traits >::pubsetbuf (char_type *
__s, streamsize __n) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubf00` dispatch to the protected derived `f00` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.71.4.27 `template<typename _CharT, typename _Traits> int
std::basic_streambuf< _CharT, _Traits >::pubsync () [inline,
inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 959

5.71.4.28 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

5.71.4.29 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.71.4.30 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf<_CharT, _Traits>::seekpos(pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file streambuf.

5.71.4.31 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT,_Traits >::setbuf(char_type *, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file streambuf.

5.71.4.32 `template<typename _CharT , typename _Traits > basic_filebuf<
_CharT, _Traits >::__streambuf_type * std::basic_filebuf<
_CharT, _Traits >::setbuf(char_type * __s, streamsize __n)
[protected, virtual, inherited]`

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 657 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::__M_buf`, `std::basic_filebuf< _CharT, _Traits >::__M_buf_size`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 961

5.71.4.33 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setg (char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.

gnext A pointer.

gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

5.71.4.34 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setp (char_type *
__pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

5.71.4.35 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sgetc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.71.4.36 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sgetn (char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.71.4.37 `template<typename _CharT, typename _Traits> streamsize
std::basic_filebuf<_CharT, _Traits>::showmanyc ()
[protected, virtual, inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 963

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 178 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::ios_base::binary`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

5.71.4.38 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::snextc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or `eof`.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.71.4.39 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc(char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

`c` The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but `c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `c`.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

5.71.4.40 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::putc (char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

5.71.4.41 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sputn (const char_type *
__s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write *s*[0] through *s*[*n*-1] to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.71.4.42 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::stoss () [inline,
inherited]`

Tosses a character.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 965

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.71.4.43 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

5.71.4.44 `template<typename _CharT, typename _Traits> int
std::basic_filebuf<_CharT, _Traits>::sync () [protected,
virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 838 of file `fstream.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.71.4.45 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 678 of file `streambuf`.

5.71.4.46 `template<typename _CharT , typename _Traits > basic_filebuf<
_CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits
>::underflow () [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 204 of file `fstream.tcc`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 967

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in()`, `std::ios_base::in`, and `std::min()`.

5.71.4.47 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::xsgetn (char_type * __s,
streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills *s*[0] through *s*[*n*-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

5.71.4.48 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::xsputn (const
char_type * __s, streamsize __n) [protected, virtual,
inherited]`

Multiple character insertion.

Parameters

s A buffer area.

n Maximum number of characters to write.

Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.71.5 Member Data Documentation

5.71.5.1 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf` [protected, inherited]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`.

5.71.5.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

5.71.5.3 `template<typename _CharT, typename _Traits> size_t std::basic_filebuf<_CharT, _Traits>::_M_buf_size` [protected, inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 969

Definition at line 116 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.5.4 `template<typename _CharT, typename _Traits> char*
std::basic_filebuf<_CharT, _Traits>::_M_ext_buf` **[protected,
inherited]**

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to [eback\(\)](#).

Definition at line 151 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.5.5 `template<typename _CharT, typename _Traits> streamsize
std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size` **[protected, inherited]**

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.5.6 `template<typename _CharT, typename _Traits> const char*
std::basic_filebuf<_CharT, _Traits>::_M_ext_next` **[protected,
inherited]**

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to [egptr\(\)](#).

Definition at line 163 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_beg` **[protected, inherited]**

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.71.5.8 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_cur
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.71.5.9 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_end
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.71.5.10 `template<typename _CharT, typename _Traits> ios_base::openmode
std::basic_filebuf< _CharT, _Traits >::_M_mode [protected,
inherited]`

Place to stash in || out || in | out settings for current filebuf.

5.71 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference 971

Definition at line 94 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.5.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file streambuf.

5.71.5.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file streambuf.

5.71.5.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

5.71.5.14 `template<typename _CharT, typename _Traits> char_type
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected,
inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 137 of file fstream.

5.71.5.15 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save
[protected, inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

5.71.5.16 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save
[protected, inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 139 of file fstream.

5.71.5.17 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf<_CharT, _Traits>::_M_pback_init
[protected, inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

5.71.5.18 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf<_CharT, _Traits>::_M_reading [protected,
inherited]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

The documentation for this class was generated from the following file:

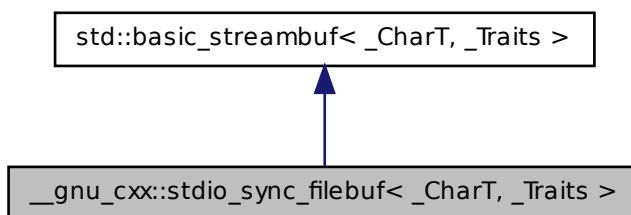
- [stdio_filebuf.h](#)

5.72 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C `FILE*`'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `traits_type::int_type` [int_type](#)
- typedef `traits_type::off_type` [off_type](#)
- typedef `traits_type::pos_type` [pos_type](#)
- typedef `_Traits` [traits_type](#)
- typedef `basic_streambuf< char_type, traits_type >` [__streambuf_type](#)

Public Member Functions

- **stdio_sync_filebuf** (`std::__c_file *__f`)
- `std::__c_file *const` [file](#) ()
- `streamsize` [in_avail](#) ()
- [int_type](#) [sbumpc](#) ()
- [int_type](#) [sgetc](#) ()
- `streamsize` [sgetn](#) ([char_type](#) *__s, `streamsize` __n)
- [int_type](#) [snextc](#) ()
- [int_type](#) [sputbackc](#) ([char_type](#) __c)
- [int_type](#) [sputc](#) ([char_type](#) __c)
- `streamsize` [sputn](#) (const [char_type](#) *__s, `streamsize` __n)
- void [stoss](#) ()
- [int_type](#) [sungetc](#) ()

Protected Member Functions

- void `gbump` (int __n)
- virtual void `imbue` (const locale &)
- virtual `int_type overflow` (`int_type` __c=traits_type::eof())
- virtual `int_type pbackfail` (`int_type` __c=traits_type::eof())
- void `pbump` (int __n)
- virtual `std::streampos seekoff` (`std::streamoff` __off, `std::ios_base::seekdir` __dir, `std::ios_base::openmode`=`std::ios_base::in`|`std::ios_base::out`)
- virtual `pos_type seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode`=`ios_base::in`|`ios_base::out`)
- virtual `std::streampos seekpos` (`std::streampos` __pos, `std::ios_base::openmode` __mode=`std::ios_base::in`|`std::ios_base::out`)
- virtual `pos_type seekpos` (`pos_type`, `ios_base::openmode`=`ios_base::in`|`ios_base::out`)
- virtual `basic_streambuf<char_type, _Traits> * setbuf` (`char_type` *, `streamsize`)
- void `setg` (`char_type` *__gbeg, `char_type` *__gnext, `char_type` *__gend)
- void `setp` (`char_type` *__pbeg, `char_type` *__pend)
- virtual `streamsize showmanyc` ()
- virtual `int sync` ()
- template<>
 `stdio_sync_filebuf<wchar_t>::int_type syncgetc` ()
- template<>
 `stdio_sync_filebuf<char>::int_type syncgetc` ()
- `int_type syncgetc` ()
- template<>
 `stdio_sync_filebuf<wchar_t>::int_type syncputc` (`int_type` __c)
- template<>
 `stdio_sync_filebuf<char>::int_type syncputc` (`int_type` __c)
- `int_type syncputc` (`int_type` __c)
- template<>
 `stdio_sync_filebuf<wchar_t>::int_type syncungetc` (`int_type` __c)
- `int_type syncungetc` (`int_type` __c)
- template<>
 `stdio_sync_filebuf<char>::int_type syncungetc` (`int_type` __c)
- virtual `int_type uflow` ()
- virtual `int_type underflow` ()
- template<>
 `std::streamsize xsgetn` (`char` *__s, `std::streamsize` __n)
- virtual `std::streamsize xsgetn` (`char_type` *__s, `std::streamsize` __n)
- template<>
 `std::streamsize xsgetn` (`wchar_t` *__s, `std::streamsize` __n)
- template<>
 `std::streamsize xspu` (const `char` *__s, `std::streamsize` __n)

- `template<>`
`std::streamsize xspn (const wchar_t * __s, std::streamsize __n)`
- `virtual std::streamsize xspn (const char_type * __s, std::streamsize __n)`
- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * ep_ptr () const`

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`
`bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`
- `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf -`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
- `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
- `class ostreambuf_iterator< char_type, traits_type >`
- `locale pubimbue (const locale & __loc)`
- `locale getloc () const`
- `__streambuf_type * pubsetbuf (char_type * __s, streamsize __n)`

- [pos_type pubseekoff](#) ([off_type __off](#), [ios_base::seekdir __way](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
- [pos_type pubseekpos](#) ([pos_type __sp](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
- [int pubsync](#) ()
- [char_type * _M_in_beg](#)
- [char_type * _M_in_cur](#)
- [char_type * _M_in_end](#)
- [char_type * _M_out_beg](#)
- [char_type * _M_out_cur](#)
- [char_type * _M_out_end](#)
- [locale _M_buf_locale](#)

5.72.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE*’s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 55 of file `stdio_sync_filebuf.h`.

5.72.2 Member Typedef Documentation

5.72.2.1 `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented in [std::basic_filebuf<_CharT, _Traits>](#), [std::basic_stringbuf<_CharT, _Traits, _Alloc>](#), [std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>](#), and [std::basic_filebuf<char_type, traits_type>](#).

Definition at line 133 of file `streambuf`.

5.72.2.2 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 59 of file `stdio_sync_filebuf.h`.

5.72.2.3 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 61 of file `stdio_sync_filebuf.h`.

5.72.2.4 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::off_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 63 of file `stdio_sync_filebuf.h`.

5.72.2.5 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::pos_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 62 of file `stdio_sync_filebuf.h`.

5.72.2.6 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _Traits __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 60 of file `stdio_sync_filebuf.h`.

5.72.3 Member Function Documentation

5.72.3.1 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback () const`
[inline, protected, inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.72.3.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr () const`
[inline, protected, inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.72.3.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.72.3.4 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> std::__c_file* const
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file ()
[inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 87 of file `stdio_sync_filebuf.h`.

5.72.3.5 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

5.72.3.6 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

5.72.3.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::gptr () const [inline,
protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.72.3.8 `template<typename _CharT, typename _Traits> virtual void
std::basic_streambuf<_CharT, _Traits>::imbue (const locale &)
[inline, protected, virtual, inherited]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in [std::basic_filebuf<_CharT, _Traits>](#), [std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>](#), and [std::basic_filebuf<char_type, traits_type>](#).

Definition at line 554 of file streambuf.

5.72.3.9 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::in_avail () [inline,
inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file streambuf.

5.72.3.10 `template<typename _CharT, typename _Traits
= std::char_traits<_CharT>> virtual int_type
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::overflow (
int_type = traits_type::eof()) [inline, protected,
virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

eof() to indicate failure, something else (usually *c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 140 of file `stdio_sync_filebuf.h`.

```
5.72.3.11  template<typename _CharT , typename _Traits
            = std::char_traits<_CharT>> virtual int_type
            _gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pbackfail (
                int_type = traits_type::eof() ) [inline, protected,
                virtual]
```

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 115 of file `stdio_sync_filebuf.h`.

5.72.3.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

5.72.3.13 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.72.3.14 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.72.3.15 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

5.72.3.16 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf<_CharT, _Traits>::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

5.72.3.17 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.72.3.18 `template<typename _CharT, typename _Traits> __streambuf_type*
std::basic_streambuf< _CharT, _Traits >::pubsetbuf (char_type *
__s, streamsize __n) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubf00` dispatch to the protected derived `f00` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.72.3.19 `template<typename _CharT, typename _Traits> int
std::basic_streambuf< _CharT, _Traits >::pubsync () [inline,
inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

5.72.3.20 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sbumpc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

5.72.3.21 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.72.3.22 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf<_CharT, _Traits>::seekpos(pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file streambuf.

```
5.72.3.23  template<typename _CharT, typename _Traits> virtual
             basic_streambuf<char_type,_Traits>* std::basic_streambuf<
             _CharT,_Traits >::setbuf( char_type *, streamsize ) [inline,
             protected, virtual, inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file streambuf.

```
5.72.3.24  template<typename _CharT, typename _Traits> void
             std::basic_streambuf<_CharT,_Traits >::setg( char_type *
             __gbeg, char_type * __gnext, char_type * __gend ) [inline,
             protected, inherited]
```

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

```
5.72.3.25  template<typename _CharT, typename _Traits> void
             std::basic_streambuf<_CharT,_Traits >::setp( char_type *
             __pbeg, char_type * __pend ) [inline, protected,
             inherited]
```

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

5.72.3.26 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sgetc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.72.3.27 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sgetn(char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.72.3.28 `template<typename _CharT, typename _Traits> virtual streamsize
std::basic_streambuf< _CharT, _Traits >::showmanyc ()
[inline, protected, virtual, inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 627 of file `streambuf`.

5.72.3.29 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::snextc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.72.3.30 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc (char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*.
If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

5.72.3.31 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc (char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

5.72.3.32 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sputn (const char_type *
__s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.72.3.33 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::stossc () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.72.3.34 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `ebackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::unget()`.

5.72.3.35 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::sync(void) [inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 159 of file `stdio_sync_filebuf.h`.

5.72.3.36 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow() [inline, protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 107 of file `stdio_sync_filebuf.h`.

5.72.3.37 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow() [inline, protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 100 of file `stdio_sync_filebuf.h`.

```
5.72.3.38 template<typename _CharT , typename _Traits =
std::char_traits<_CharT>> virtual std::streamsize
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xsgetn (
char_type * __s, std::streamsize __n ) [protected, virtual]
```

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by [sbumpc\(\)](#). Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

```
5.72.3.39 template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> virtual std::streamsize
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xsputn ( const
char_type * __s, std::streamsize __n ) [protected, virtual]
```

Multiple character insertion.

Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

5.72.4 Member Data Documentation

```
5.72.4.1 template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale
[protected, inherited]
```

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

```
5.72.4.2 template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_beg
[protected, inherited]
```

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.72.4.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.72.4.4 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.72.4.5 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.72.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.72.4.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_out_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

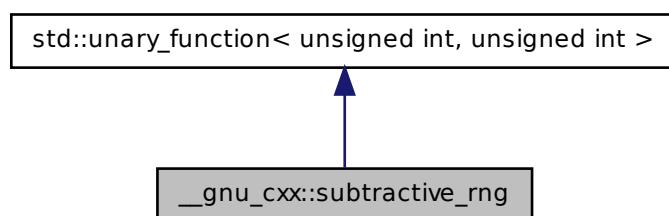
Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following file:

- [stdio_sync_filebuf.h](#)

5.73 `__gnu_cxx::subtractive_rng` Class Reference

Inheritance diagram for `__gnu_cxx::subtractive_rng`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- [subtractive_rng](#) (unsigned int `__seed`)
- [subtractive_rng](#) ()
- void `_M_initialize` (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

5.73.1 Detailed Description

The [subtractive_rng](#) class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

Definition at line 347 of file `ext/functional`.

5.73.2 Member Typedef Documentation

5.73.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.73.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

5.73.3 Constructor & Destructor Documentation

5.73.3.1 `__gnu_cxx::subtractive_rng::subtractive_rng (unsigned int __seed)
[inline]`

Ctor allowing you to initialize the seed.

Definition at line 389 of file `ext/functional`.

5.73.3.2 `__gnu_cxx::subtractive_rng::subtractive_rng () [inline]`

Default ctor; initializes its state with some number you don't see.

Definition at line 393 of file `ext/functional`.

5.73.4 Member Function Documentation

5.73.4.1 `unsigned int __gnu_cxx::subtractive_rng::operator() (unsigned int
__limit) [inline]`

Returns a number less than the argument.

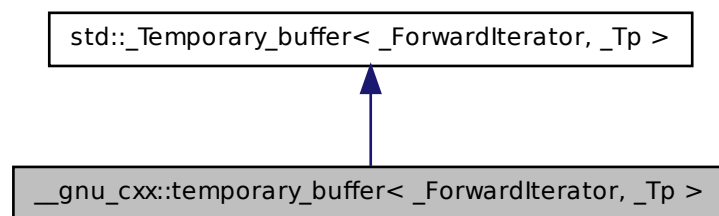
Definition at line 358 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

5.74 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [temporary_buffer](#) (_ForwardIterator __first, _ForwardIterator __last)
- [~temporary_buffer](#) ()
- iterator [begin](#) ()
- iterator [end](#) ()
- size_type [requested_size](#) () const
- size_type [size](#) () const

Protected Attributes

- pointer **_M_buffer**
- size_type **_M_len**
- size_type **_M_original_len**

5.74.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type> struct __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>
```

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

Definition at line 182 of file `ext/memory`.

5.74.2 Constructor & Destructor Documentation

```
5.74.2.1 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp
>::temporary_buffer ( _ForwardIterator __first, _ForwardIterator
__last ) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

Definition at line 185 of file `ext/memory`.

```
5.74.2.2 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp
>::~~temporary_buffer ( ) [inline]
```

Destroys objects and frees storage.

Definition at line 189 of file `ext/memory`.

5.74.3 Member Function Documentation

5.74.3.1 `template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()
[inline, inherited]`

As per Table mumble.

Definition at line 150 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

5.74.3.2 `template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()
[inline, inherited]`

As per Table mumble.

Definition at line 155 of file `stl_tempbuf.h`.

5.74.3.3 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size () const [inline, inherited]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 145 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

5.74.3.4 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const
[inline, inherited]`

As per Table mumble.

Definition at line 140 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

5.75 `__gnu_cxx::throw_allocator_base< _Tp, _Cond >` > Class Template Reference

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

Inheritance diagram for `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`:



Public Types

- typedef `const value_type *` **const_pointer**
- typedef `const value_type &` **const_reference**
- typedef `ptrdiff_t` **difference_type**
- typedef `value_type *` **pointer**
- typedef `value_type &` **reference**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- pointer **address** (reference `__x`) const
- `const_pointer` **address** (const_reference `__x`) const
- pointer **allocate** (size_type `__n`, [std::allocator< void >::const_pointer](#) hint=0)
- void **check_allocated** (pointer `__p`, size_type `__n`)
- void **check_allocated** (size_type `__n`)
- void **check_allocated** (void *`p`, size_t size)
- void **check_allocated** (size_t label)
- void **construct** (pointer `__p`, const value_type &val)
- template<typename... `_Args`>
void **construct** (pointer `__p`, `_Args` &&...`__args`)
- void **deallocate** (pointer `__p`, size_type `__n`)
- void **destroy** (pointer `__p`)
- void **erase** (void *`p`, size_t size)
- void **insert** (void *`p`, size_t size)
- size_type **max_size** () const throw ()

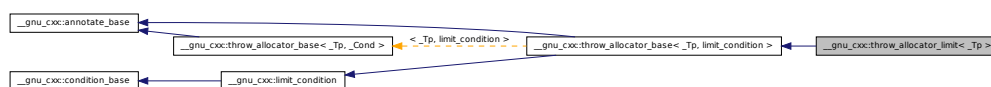
- static size_t **get_label** ()
- static void **set_label** (size_t l)

```
template<typename _Tp, typename _Cond> class __gnu_cxx::throw_allocator_-  
base< _Tp, _Cond >
```

Definition at line 598 of file throw_allocator.h.

- [throw_allocator.h](#)

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_limit** (const [throw_allocator_limit](#) &) throw ()
- `template<typename _Tp1 >`
throw_allocator_limit (const [throw_allocator_limit](#)<_Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check_allocated** (size_t label)
- void **check_allocated** (size_type __n)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (void *p, size_t size)
- void **construct** (pointer __p, const value_type &val)
- void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (pointer __p)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)
- size_type **max_size** () const throw ()

Static Public Member Functions

- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

5.76.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_limit<_Tp>`

Allocator throwing via limit condition.

Definition at line 688 of file `throw_allocator.h`.

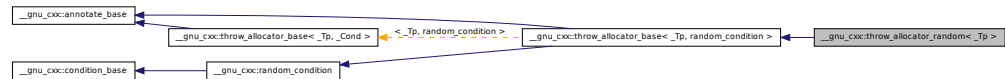
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.77 `__gnu_cxx::throw_allocator_random< _Tp >` Struct Template Reference

Allocator throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_allocator_random< _Tp >`:



Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `throw_allocator_random` (const `throw_allocator_random` &) `throw ()`
- `template<typename _Tp1 >`
`throw_allocator_random` (const `throw_allocator_random< _Tp1 >` &) `throw ()`
- pointer `address` (reference `__x`) `const`
- `const_pointer` `address` (const_reference `__x`) `const`
- pointer `allocate` (size_type `__n`, `std::allocator< void >::const_pointer` hint=0)
- void `check_allocated` (size_t label)
- void `check_allocated` (void *p, size_t size)
- void `check_allocated` (pointer `__p`, size_type `__n`)
- void `check_allocated` (size_type `__n`)
- void `construct` (pointer `__p`, const value_type &val)
- void `construct` (pointer `__p`, _Args &&... __args)
- void `deallocate` (pointer `__p`, size_type `__n`)
- void `destroy` (pointer `__p`)
- void `erase` (void *p, size_t size)
- void `insert` (void *p, size_t size)

5.78 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference 1007

- `size_type` **max_size** () const throw ()
- void **seed** (unsigned long __s)

Static Public Member Functions

- static `size_t` **get_label** ()
- static void **set_label** (`size_t` l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

5.77.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_random<_Tp>`

Allocator throwing via random condition.

Definition at line 707 of file `throw_allocator.h`.

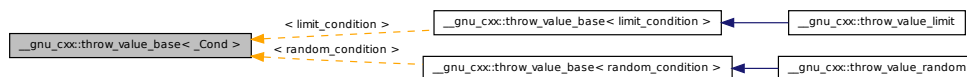
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.78 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Class with exception generation control. Intended to be used as a `value_type` in templated code.

Inheritance diagram for `__gnu_cxx::throw_value_base<_Cond>`:



Public Types

- `typedef _Cond` **condition_type**

Public Member Functions

- **throw_value_base** (const [throw_value_base](#) &__v)
- **throw_value_base** (const std::size_t __i)
- [throw_value_base](#) & **operator++** ()
- [throw_value_base](#) & **operator=** (const [throw_value_base](#) &__v)

Public Attributes

- std::size_t **_M_i**

5.78.1 Detailed Description

template<typename _Cond> struct __gnu_cxx::throw_value_base< _Cond >

Class with exception generation control. Intended to be used as a value_type in templated code. Note: Destructor not allowed to throw.

Definition at line 453 of file [throw_allocator.h](#).

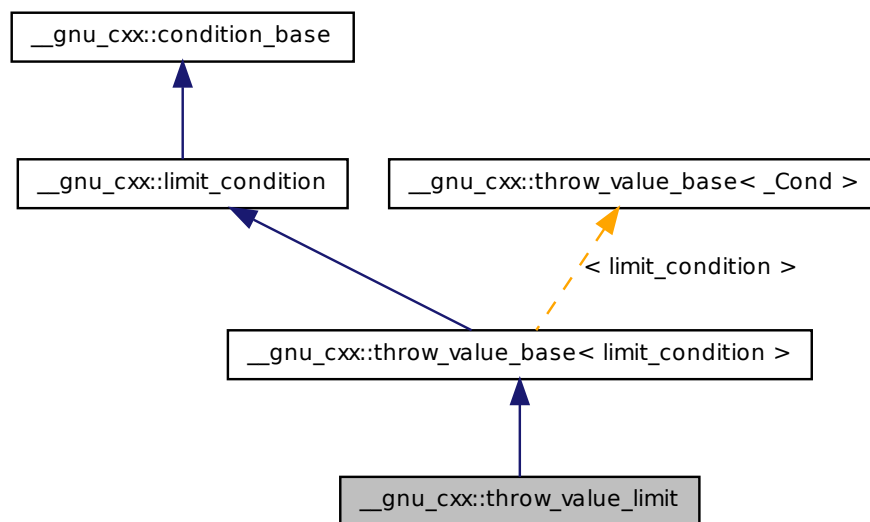
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.79 __gnu_cxx::throw_value_limit Struct Reference

Type throwing via limit condition.

Inheritance diagram for `__gnu_cxx::throw_value_limit`:



Public Types

- typedef `throw_value_base< limit_condition > base_type`
- typedef `limit_condition condition_type`

Public Member Functions

- `throw_value_limit` (const `throw_value_limit` &__other)
- `throw_value_limit` (const std::size_t __i)
- `throw_value_base` & `operator++` ()

Static Public Member Functions

- static size_t & `count` ()
- static size_t & `limit` ()
- static void `set_limit` (const size_t __l)
- static void `throw_conditionally` ()

Public Attributes

- `std::size_t _M_i`

5.79.1 Detailed Description

Type throwing via limit condition.

Definition at line 559 of file `throw_allocator.h`.

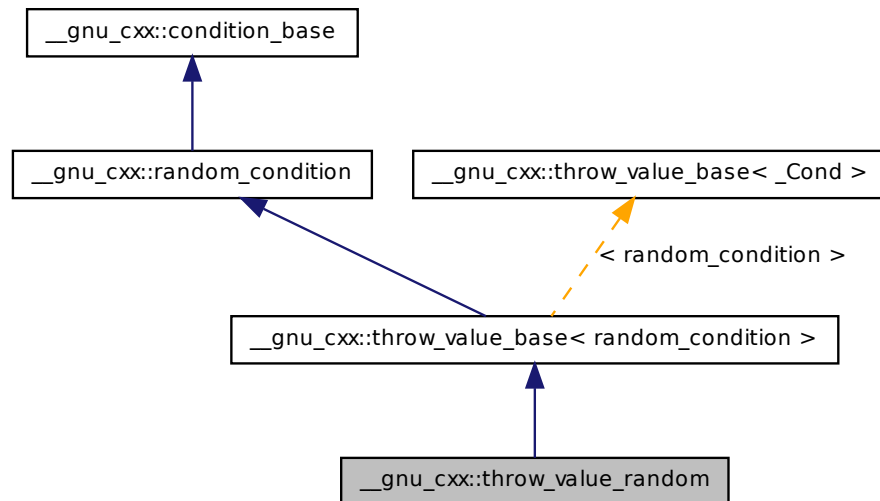
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.80 `__gnu_cxx::throw_value_random` Struct Reference

Type throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_value_random`:



Public Types

- typedef [throw_value_base< random_condition >](#) **base_type**
- typedef [random_condition](#) **condition_type**

Public Member Functions

- **throw_value_random** (const [throw_value_random](#) &__other)
- **throw_value_random** (const std::size_t __i)
- [throw_value_base](#) & **operator++** ()
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**

5.80.1 Detailed Description

Type throwing via random condition.

Definition at line 574 of file `throw_allocator.h`.

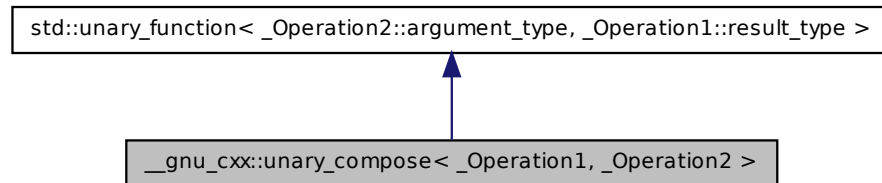
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.81 `__gnu_cxx::unary_compose< _Operation1, _Operation2 >` Class Template Reference

An [SGI extension](#) .

Inheritance diagram for `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **unary_compose** (const `_Operation1` &__x, const `_Operation2` &__y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &__x) const

Protected Attributes

- `_Operation1` **_M_fn1**
- `_Operation2` **_M_fn2**

5.81.1 Detailed Description

`template<class _Operation1, class _Operation2> class __gnu_cxx::unary_compose< _Operation1, _Operation2 >`

An [SGI extension](#) .

Definition at line 124 of file `ext/functional`.

5.82 `__gnu_debug::__is_same< _Type1, _Type2 >` Struct Template Reference

5.81.2 Member Typedef Documentation

5.81.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.81.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

5.82 `__gnu_debug::__is_same< _Type1, _Type2 >` Struct Template Reference

Static Public Attributes

- static const bool `value`

5.82.1 Detailed Description

`template<typename _Type1, typename _Type2> struct __gnu_debug::__is_
same< _Type1, _Type2 >`

Determine if the two types are the same.

Definition at line 43 of file `formatter.h`.

The documentation for this struct was generated from the following file:

- [formatter.h](#)

5.83 `__gnu_debug::_After_nth_from< _Iterator >` Class Template Reference

Public Member Functions

- `_After_nth_from` (const difference_type &__n, const _Iterator &__base)
- `bool operator()` (const _Iterator &__x) const

5.83.1 Detailed Description

```
template<typename _Iterator> class __gnu_debug::_After_nth_from< _Iterator
>
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 63 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

5.84 `__gnu_debug::_Not_equal_to< _Type >` Class Template Reference

Public Member Functions

- `_Not_equal_to` (const _Type &__v)
- `bool operator()` (const _Type &__x) const

5.84.1 Detailed Description

```
template<typename _Type> class __gnu_debug::_Not_equal_to< _Type >
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

Definition at line 48 of file `safe_sequence.h`.

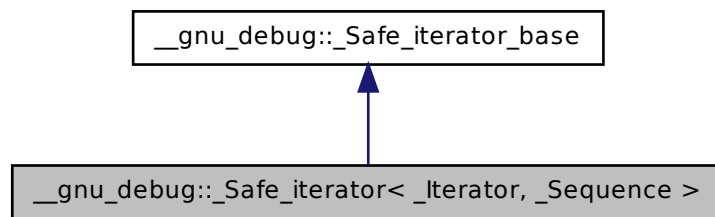
The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

5.85 `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >` Class Template Reference

Safe iterator wrapper.

Inheritance diagram for `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >`:



Public Types

- `typedef _Iterator Base_iterator`
- `typedef _Traits::difference_type difference_type`
- `typedef _Traits::iterator_category iterator_category`
- `typedef _Traits::pointer pointer`
- `typedef _Traits::reference reference`
- `typedef _Traits::value_type value_type`

Public Member Functions

- `_Safe_iterator ()`
- `_Safe_iterator (const _Iterator &__i, const _Sequence *__seq)`
- `template<typename _MutableIterator >
_Safe_iterator (const _Safe_iterator< _MutableIterator, typename __gnu_cxx::__enable_if<(std::__are_same< _MutableIterator, typename _Sequence::iterator::_Base_iterator >::__value), _Sequence >::__type > &__x)`
- `_Safe_iterator (const _Safe_iterator &__x)`
- `void _M_attach (const _Sequence *__seq)`
- `void _M_attach (_Safe_sequence_base *__seq, bool __constant)`

- void [_M_attach_single](#) ([_Safe_sequence_base](#) *__seq, bool __constant) throw ()
- void [_M_attach_single](#) (const [_Sequence](#) *__seq)
- bool [_M_attached_to](#) (const [_Safe_sequence_base](#) *__seq) const
- bool [_M_can_advance](#) (const difference_type &__n) const
- [_GLIBCXX_PURE](#) bool [_M_can_compare](#) (const [_Safe_iterator_base](#) &__x) const throw ()
- bool [_M_decrementable](#) () const
- bool [_M_dereferenceable](#) () const
- void [_M_detach](#) ()
- void [_M_detach_single](#) () throw ()
- const [_Sequence](#) * [_M_get_sequence](#) () const
- bool [_M_incrementable](#) () const
- void [_M_invalidate](#) ()
- void [_M_invalidate_single](#) ()
- bool [_M_is_begin](#) () const
- bool [_M_is_end](#) () const
- [_GLIBCXX_PURE](#) bool [_M_singular](#) () const throw ()
- template<typename [_Other](#) >
bool [_M_valid_range](#) (const [_Safe_iterator](#)< [_Other](#), [_Sequence](#) > &__rhs) const
- [_Iterator](#) base () const
- operator [_Iterator](#) () const
- reference operator* () const
- [_Safe_iterator](#) operator+ (const difference_type &__n) const
- [_Safe_iterator](#) & operator++ ()
- [_Safe_iterator](#) operator++ (int)
- [_Safe_iterator](#) & operator+= (const difference_type &__n)
- [_Safe_iterator](#) operator- (const difference_type &__n) const
- [_Safe_iterator](#) operator-- (int)
- [_Safe_iterator](#) & operator-- ()
- [_Safe_iterator](#) & operator-= (const difference_type &__n)
- pointer operator-> () const
- [_Safe_iterator](#) & operator= (const [_Safe_iterator](#) &__x)
- reference operator[] (const difference_type &__n) const

Static Public Member Functions

- template<typename [_Iterator1](#) , typename [_Iterator2](#) >
static [std::pair](#)< difference_type, [_Distance_precision](#) > [_M_get_distance](#)
(const [_Iterator1](#) &__lhs, const [_Iterator2](#) &__rhs)
- template<typename [_Iterator1](#) , typename [_Iterator2](#) >
static [std::pair](#)< difference_type, [_Distance_precision](#) > [_M_get_distance](#)
(const [_Iterator1](#) &__lhs, const [_Iterator2](#) &__rhs, [std::forward_iterator_tag](#))

5.85 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1017

- `template<typename _Iterator1, typename _Iterator2 >`
 static `std::pair< difference_type, _Distance_precision > _M_get_distance`
 (const `_Iterator1` &__lhs, const `_Iterator2` &__rhs, `std::random_access_iterator_tag`)

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- unsigned int `_M_version`

Protected Member Functions

- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

5.85.1 Detailed Description

`template<typename _Iterator, typename _Sequence> class __gnu_debug::_Safe_iterator<_Iterator, _Sequence>`

Safe iterator wrapper. The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 63 of file `safe_iterator.h`.

5.85.2 Constructor & Destructor Documentation

5.85.2.1 `template<typename _Iterator, typename _Sequence>`
 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator`
 () [`inline`]

Postcondition

the iterator is singular and unattached

Definition at line 99 of file `safe_iterator.h`.

5.85.2.2 `template<typename _Iterator, typename _Sequence>
 __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator
 (const _Iterator & __i, const _Sequence * __seq) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 108 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

5.85.2.3 `template<typename _Iterator, typename _Sequence>
 __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator
 (const _Safe_iterator< _Iterator, _Sequence > & __x) [inline]`

Copy construction.

Definition at line 119 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

5.85.2.4 `template<typename _Iterator, typename _Sequence>
 template<typename _MutableIterator > __gnu_debug::_-
 Safe_iterator< _Iterator, _Sequence >::_Safe_iterator
 (const _Safe_iterator< _MutableIterator, typename
 __gnu_cxx::enable_if<(std::_are_same< _MutableIterator,
 typename _Sequence::iterator::_Base_iterator >::_value), _Sequence
 >::_type > & __x) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 136 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

5.85.3 Member Function Documentation

5.85.3.1 `template<typename _Iterator, typename _Sequence> void
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach(
const _Sequence * __seq) [inline]`

Attach iterator to the given sequence.

Definition at line 321 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator=()`.

5.85.3.2 `void __gnu_debug::_Safe_iterator_base::_M_attach(
_Safe_sequence_base * __seq, bool __constant) [inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`.

5.85.3.3 `template<typename _Iterator, typename _Sequence>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence>
>::_M_attach_single(const _Sequence * __seq) [inline]`

Likewise, but not thread-safe.

Definition at line 329 of file `safe_iterator.h`.

5.85.3.4 `void __gnu_debug::_Safe_iterator_base::_M_attach_single
(_Safe_sequence_base * __seq, bool __constant) throw ()
[inherited]`

Likewise, but not thread-safe.

5.85.3.5 `bool __gnu_debug::_Safe_iterator_base::_M_attached_to(const
_Safe_sequence_base * __seq) const [inline, inherited]`

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

5.85.3.6 `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw ()`
[inherited]

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

5.85.3.7 `template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::_M_dereferenceable () const` **[inline]**

Is the iterator dereferenceable?

Definition at line 345 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_check_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator*()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`.

5.85.3.8 `void __gnu_debug::_Safe_iterator_base::_M_detach ()`
[inherited]

Detach the iterator for whatever sequence it is attached to, if any.

5.85.3.9 `void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw ()` **[inherited]**

Likewise, but not thread-safe.

5.85.3.10 `template<typename _Iterator, typename _Sequence>
template<typename _Iterator1, typename _Iterator2 >
static std::pair<difference_type, _Distance_precision>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::_M_get_distance (const _Iterator1 & __lhs, const _Iterator2 &
__rhs)` **[inline, static]**

Determine the distance between two iterators with some known precision.

Definition at line 375 of file `safe_iterator.h`.

5.85 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1021

5.85.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex () throw ()` `[protected, inherited]`

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate()`.

5.85.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_incrementable () const` `[inline]`

Is the iterator incrementable?

Definition at line 350 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++()`.

5.85.3.13 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate ()`

Invalidate the iterator, making it singular.

Definition at line 106 of file `safe_iterator.tcc`.

References `__gnu_debug::Safe_iterator_base::M_get_mutex()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`.

5.85.3.14 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single ()`

Likewise, but not thread-safe.

Definition at line 115 of file `safe_iterator.tcc`.

References `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_next`, `__gnu_debug::Safe_iterator_base::M_sequence`, `__gnu_debug::Safe_iterator_base::M_singular()`, `__gnu_debug::Safe_iterator_base::M_version`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

5.85.3.15 `template<typename _Iterator, typename _Sequence> bool
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin
() const [inline]`

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 400 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

5.85.3.16 `template<typename _Iterator, typename _Sequence> bool
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end () const [inline]`

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 404 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

5.85.3.17 `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw () [inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular()`, `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.85.3.18 `template<typename _Iterator, typename _Sequence> _Iterator
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base () const [inline]`

Return the underlying iterator.

Definition at line 311 of file `safe_iterator.h`.

5.85 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1023

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.85.3.19 `template<typename _Iterator, typename _Sequence>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator
_Iterator () const [inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 317 of file `safe_iterator.h`.

5.85.3.20 `template<typename _Iterator, typename _Sequence> reference
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator* () const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 175 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

5.85.3.21 `template<typename _Iterator, typename _Sequence> _Safe_iterator
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ (int) [inline]`

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 218 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

5.85.3.22 `template<typename _Iterator, typename _Sequence>
 _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence
 >::operator++ () [inline]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 204 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _-
 Iterator, _Sequence >::_M_incrementable()`.

5.85.3.23 `template<typename _Iterator, typename _Sequence> _Safe_iterator
 __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- (
 int) [inline]`

Iterator postdecrement.

Precondition

iterator is decrementable

Definition at line 248 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

5.85.3.24 `template<typename _Iterator, typename _Sequence>
 _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence
 >::operator-- () [inline]`

Iterator predecrement.

Precondition

iterator is decrementable

Definition at line 234 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

5.85.3.25 `template<typename _Iterator, typename _Sequence> pointer
 __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> (
) const [inline]`

Iterator dereference.

5.85 __gnu_debug::_Safe_iterator< _Iterator, _Sequence > Class Template Reference 1025

Precondition

iterator is dereferenceable

Todo

Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Definition at line 190 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

```
5.85.3.26 template<typename _Iterator, typename _Sequence>
    _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence
    >::operator=( const _Safe_iterator< _Iterator, _Sequence > & __x
    ) [inline]
```

Copy assignment.

Definition at line 156 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach()`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

5.85.4 Member Data Documentation

5.85.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next`
[**inherited**]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

5.85.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior`
[**inherited**]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

5.85.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` **[inherited]**

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`, `__gnu_debug::_Safe_iterator_base::_M_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.85.4.4 `unsigned int __gnu_debug::_Safe_iterator_base::_M_version` **[inherited]**

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

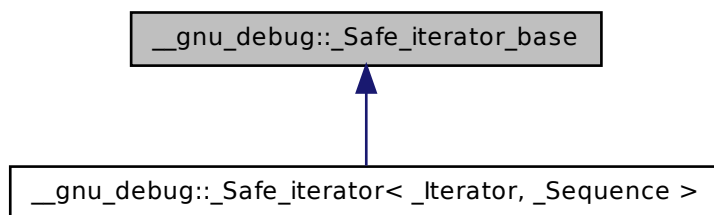
The documentation for this class was generated from the following files:

- [safe_iterator.h](#)
- [safe_iterator.tcc](#)

5.86 `__gnu_debug::_Safe_iterator_base` Class Reference

Basic functionality for a *safe* iterator.

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- void `_M_attach` (`_Safe_sequence_base *__seq`, bool `__constant`)
- void `_M_attach_single` (`_Safe_sequence_base *__seq`, bool `__constant`) throw ()
- bool `_M_attached_to` (const `_Safe_sequence_base *__seq`) const
- `_GLIBCXX_PURE` bool `_M_can_compare` (const `_Safe_iterator_base &__x`) const throw ()
- void `_M_detach` ()
- void `_M_detach_single` () throw ()
- `_GLIBCXX_PURE` bool `_M_singular` () const throw ()

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- unsigned int `_M_version`

Protected Member Functions

- `_Safe_iterator_base` ()
- `_Safe_iterator_base` (const `_Safe_sequence_base *__seq`, bool `__constant`)
- `_Safe_iterator_base` (const `_Safe_iterator_base &`)
- `_Safe_iterator_base` (const `_Safe_iterator_base &__x`, bool `__constant`)
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `_Safe_iterator_base & operator=` (const `_Safe_iterator_base &`)

5.86.1 Detailed Description

Basic functionality for a *safe* iterator. The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 51 of file `safe_base.h`.

5.86.2 Constructor & Destructor Documentation

5.86.2.1 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ()` [`inline`, `protected`]

Initializes the iterator and makes it singular.

Definition at line 77 of file `safe_base.h`.

5.86.2.2 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant)` [`inline`, `protected`]

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 88 of file `safe_base.h`.

References `_M_attach()`.

5.86.2.3 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant)` [`inline`, `protected`]

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 95 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.

5.86.3 Member Function Documentation

5.86.3.1 void __gnu_debug::_Safe_iterator_base::_M_attach (
_Safe_sequence_base * __seq, bool __constant)

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by _Safe_iterator_base().

5.86.3.2 void __gnu_debug::_Safe_iterator_base::_M_attach_single (
_Safe_sequence_base * __seq, bool __constant) throw ()

Likewise, but not thread-safe.

5.86.3.3 bool __gnu_debug::_Safe_iterator_base::_M_attached_to (const
_Safe_sequence_base * __seq) const [inline]

Determines if we are attached to the given sequence.

Definition at line 130 of file safe_base.h.

References _M_sequence.

5.86.3.4 _GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_
can_compare (const _Safe_iterator_base & __x) const throw
()

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.

5.86.3.5 void __gnu_debug::_Safe_iterator_base::_M_detach ()

Detach the iterator for whatever sequence it is attached to, if any.

5.86.3.6 void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw
()

Likewise, but not thread-safe.

5.86.3.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected]`

For use in [_Safe_iterator](#).

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

5.86.3.8 `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw ()`

Is this iterator singular?

Referenced by `__gnu_debug::__check_singular()`, `__gnu_debug::__check_singular_aux()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.86.4 Member Data Documentation

5.86.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

5.86.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

5.86.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence`

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

5.87 `__gnu_debug::_Safe_sequence< _Sequence >` Class Template Reference

Referenced by `_M_attached_to()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`, `_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.86.4.4 `unsigned int __gnu_debug::_Safe_iterator_base::_M_version`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

The documentation for this class was generated from the following file:

- [safe_base.h](#)

5.87 `__gnu_debug::_Safe_sequence< _Sequence >` Class Template Reference

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

Inheritance diagram for `__gnu_debug::_Safe_sequence< _Sequence >`:



Public Member Functions

- void [_M_invalidate_all](#) () const
- template<typename [_Predicate](#) >
void [_M_invalidate_if](#) ([_Predicate](#) __pred)
- template<typename [_Iterator](#) >
void [_M_transfer_iter](#) (const [_Safe_iterator](#)< [_Iterator](#), [_Sequence](#) > &__x)

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) &__x)

5.87.1 Detailed Description

```
template<typename \_Sequence> class \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it. The class template [_Safe_sequence](#) simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and types that are instantiations of class template [_Safe_iterator](#) for this sequence. Iterators will then be tracked automatically.

Definition at line 97 of file `safe_sequence.h`.

5.87 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference 033

5.87.2 Member Function Documentation

5.87.2.1 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.87.2.2 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.87.2.3 `__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw ()` [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.87.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const`
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.87.2.5 `template<typename _Sequence> template<typename _Predicate> void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (_Predicate __pred)`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

Definition at line 121 of file `safe_sequence.h`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_get_mutex()`, and `__gnu_debug::_Safe_sequence_base::_M_iterators`.

5.87.2.6 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` `[protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.87.2.7 `void __gnu_debug::Safe_sequence_base::M_swap (` `_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.87.2.8 `template<typename _Sequence> template<typename _Iterator >` `void __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter` `(const _Safe_iterator< _Iterator, _Sequence > & __x)`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

Definition at line 154 of file `safe_sequence.h`.

References `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_sequence_base::M_get_mutex()`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_sequence`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::base()`.

5.87.3 Member Data Documentation

5.87.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M-` `const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

5.87.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M-` `iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter().

5.87.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

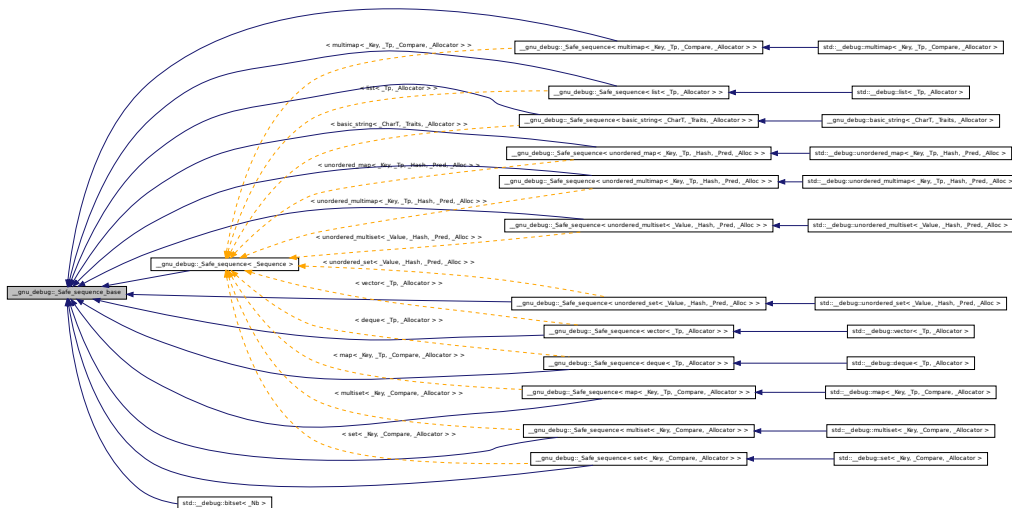
The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

5.88 __gnu_debug::_Safe_sequence_base Class Reference

Base class that supports tracking of iterators that reference a sequence.

Inheritance diagram for __gnu_debug::_Safe_sequence_base:



Public Member Functions

- void [_M_invalidate_all](#) () const

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- [~_Safe_sequence_base](#) ()
- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) &__x)

5.88.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence. The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit [_Safe_iterator_base](#)) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 159 of file `safe_base.h`.

5.88.2 Constructor & Destructor Documentation

5.88.2.1 `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base ()` [inline, protected]

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 179 of file `safe_base.h`.

5.88.3 Member Function Documentation

5.88.3.1 void __gnu_debug::_Safe_sequence_base::_M_detach_all ()
[protected]

Detach all iterators, leaving them singular.

5.88.3.2 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ()
[protected]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.88.3.3 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected]

For use in [_Safe_sequence](#).

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter().

5.88.3.4 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const
[inline]

Invalidates all iterators.

Definition at line 215 of file safe_base.h.

5.88.3.5 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ()
[protected]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.88.3.6 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.88.4 Member Data Documentation

5.88.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.88.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.88.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

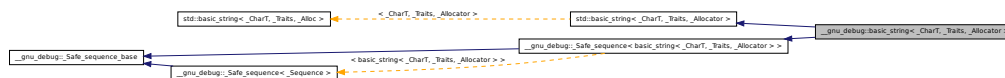
- [safe_base.h](#)

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference

Class `std::basic_string` with safety/checking/debug instrumentation.

5.89 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference 1039

Inheritance diagram for __gnu_debug::basic_string< _CharT, _Traits, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< typename _Base::const_iterator, [basic_string](#) > **const_iterator**
- typedef __gnu_cxx::__normal_iterator< const_pointer, [basic_string](#) > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _CharT_alloc_type::const_pointer **const_pointer**
- typedef _CharT_alloc_type::const_reference **const_reference**
- typedef _Base::const_reference **const_reference**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _CharT_alloc_type::difference_type **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, [basic_string](#) > **iterator**
- typedef __gnu_debug::_Safe_iterator< typename _Base::iterator, [basic_string](#) > **iterator**
- typedef _Base::pointer **pointer**
- typedef _CharT_alloc_type::pointer **pointer**
- typedef _CharT_alloc_type::reference **reference**
- typedef _Base::reference **reference**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Base::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **basic_string** (const _Allocator &__a=_Allocator())
- **basic_string** (const [_Base](#) &__base)
- **basic_string** (const [basic_string](#) &__str, size_type __pos, size_type __n=[_Base::npos](#), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
basic_string (_InputIterator __begin, _InputIterator __end, const _Allocator &__a=_Allocator())
- **basic_string** ([basic_string](#) &&__str)
- **basic_string** (const _CharT *__s, size_type __n, const _Allocator &__a=_Allocator())
- **basic_string** (std::initializer_list< _CharT > __l, const _Allocator &__a=_Allocator())
- **basic_string** (const [basic_string](#) &__str)
- **basic_string** (const _CharT *__s, const _Allocator &__a=_Allocator())
- **basic_string** (size_type __n, _CharT __c, const _Allocator &__a=_Allocator())

- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- void [_M_invalidate_all](#) () const
- void [_M_invalidate_if](#) (_Predicate __pred)
- void [_M_transfer_iter](#) (const [_Safe_iterator](#)< _Iterator, [basic_string](#)< _CharT, _Traits, _Allocator > > &__x)
- [basic_string](#) & **append** (const [basic_string](#) &__str)
- [basic_string](#) & **append** (const [basic_string](#) &__str, size_type __pos, size_type __n)
- [basic_string](#) & **append** (const [basic_string](#) &__str)
- [basic_string](#) & **append** (const _CharT *__s, size_type __n)
- [basic_string](#) & **append** (const [basic_string](#) &__str, size_type __pos, size_type __n)
- [basic_string](#) & **append** (const _CharT *__s, size_type __n)
- [basic_string](#) & **append** (const _CharT *__s)
- [basic_string](#) & **append** (size_type __n, _CharT __c)
- [basic_string](#) & **append** (const _CharT *__s)
- [basic_string](#) & **append** (initializer_list< _CharT > __l)
- [basic_string](#) & **append** (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & **append** (size_type __n, _CharT __c)
- template<typename _InputIterator >
[basic_string](#) & **append** (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & **assign** (const [basic_string](#) &__str)
- [basic_string](#) & **assign** ([basic_string](#) &&__str)
- [basic_string](#) & **assign** (const [basic_string](#) &__str, size_type __pos, size_type __n)

- `basic_string` & `assign` (const `_CharT` *`__s`, `size_type` `__n`)
- `basic_string` & `assign` (const `_CharT` *`__s`)
- `basic_string` & `assign` (`size_type` `__n`, `_CharT` `__c`)
- `basic_string` & `assign` (`_InputIterator` `__first`, `_InputIterator` `__last`)
- `basic_string` & `assign` (`initializer_list`< `_CharT` > `__l`)
- `basic_string` & `assign` (const `basic_string` &`__x`)
- `basic_string` & `assign` (`basic_string` &&`__x`)
- `basic_string` & `assign` (const `basic_string` &`__str`, `size_type` `__pos`, `size_type` `__n`)
- `basic_string` & `assign` (const `_CharT` *`__s`, `size_type` `__n`)
- `basic_string` & `assign` (const `_CharT` *`__s`)
- `basic_string` & `assign` (`size_type` `__n`, `_CharT` `__c`)
- `template<typename _InputIterator>`
 `basic_string` & `assign` (`_InputIterator` `__first`, `_InputIterator` `__last`)
- `basic_string` & `assign` (`std::initializer_list`< `_CharT` > `__l`)
- `const_reference` `at` (`size_type` `__n`) `const`
- `reference` `at` (`size_type` `__n`)
- `reference` `back` ()
- `const_reference` `back` () `const`
- `iterator` `begin` ()
- `const_iterator` `begin` () `const`
- `iterator` `begin` ()
- `const_iterator` `begin` () `const`
- `const _CharT` * `c_str` () `const`
- `const _CharT` * `c_str` () `const`
- `size_type` `capacity` () `const`
- `const_iterator` `cbegin` () `const`
- `const_iterator` `cend` () `const`
- `void` `clear` ()
- `void` `clear` ()
- `int` `compare` (`size_type` `__pos`, `size_type` `__n1`, const `_CharT` *`__s`, `size_type` `__n2`) `const`
- `int` `compare` (`size_type` `__pos`, `size_type` `__n1`, const `_CharT` *`__s`) `const`
- `int` `compare` (const `basic_string` &`__str`) `const`
- `int` `compare` (`size_type` `__pos1`, `size_type` `__n1`, const `basic_string` &`__str`) `const`
- `int` `compare` (`size_type` `__pos1`, `size_type` `__n1`, const `basic_string` &`__str`, `size_type` `__pos2`, `size_type` `__n2`) `const`
- `int` `compare` (`size_type` `__pos1`, `size_type` `__n1`, const `_CharT` *`__s`) `const`
- `int` `compare` (const `_CharT` *`__s`) `const`
- `int` `compare` (`size_type` `__pos1`, `size_type` `__n1`, const `_CharT` *`__s`, `size_type` `__n2`) `const`
- `int` `compare` (`size_type` `__pos`, `size_type` `__n`, const `basic_string` &`__str`) `const`

- int `compare` (const `_CharT *__s`) const
- int `compare` (const `basic_string` &__str) const
- int `compare` (size_type __pos1, size_type __n1, const `basic_string` &__str, size_type __pos2, size_type __n2) const
- size_type `copy` (`_CharT *__s`, size_type __n, size_type __pos=0) const
- size_type `copy` (`_CharT *__s`, size_type __n, size_type __pos=0) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- const `_CharT *` `data` () const
- const `_CharT *` `data` () const
- bool `empty` () const
- `iterator` `end` ()
- `iterator` `end` ()
- `const_iterator` `end` () const
- `const_iterator` `end` () const
- `basic_string` & `erase` (size_type __pos=0, size_type __n=`npos`)
- `iterator` `erase` (`iterator` __position)
- `iterator` `erase` (`iterator` __first, `iterator` __last)
- `basic_string` & `erase` (size_type __pos=0, size_type __n=`_Base::npos`)
- `iterator` `erase` (`iterator` __position)
- `iterator` `erase` (`iterator` __first, `iterator` __last)
- size_type `find` (const `basic_string` &__str, size_type __pos=0) const
- size_type `find` (const `basic_string` &__str, size_type __pos=0) const
- size_type `find` (const `_CharT *__s`, size_type __pos, size_type __n) const
- size_type `find` (const `_CharT *__s`, size_type __pos=0) const
- size_type `find` (`_CharT __c`, size_type __pos=0) const
- size_type `find` (const `_CharT *__s`, size_type __pos, size_type __n) const
- size_type `find` (const `_CharT *__s`, size_type __pos=0) const
- size_type `find` (`_CharT __c`, size_type __pos=0) const
- size_type `find_first_not_of` (`_CharT __c`, size_type __pos=0) const
- size_type `find_first_not_of` (const `_CharT *__s`, size_type __pos, size_type __n) const
- size_type `find_first_not_of` (`_CharT __c`, size_type __pos=0) const
- size_type `find_first_not_of` (const `_CharT *__s`, size_type __pos, size_type __n) const
- size_type `find_first_not_of` (const `_CharT *__s`, size_type __pos=0) const
- size_type `find_first_not_of` (const `basic_string` &__str, size_type __pos=0) const
- size_type `find_first_not_of` (const `basic_string` &__str, size_type __pos=0) const
- size_type `find_first_not_of` (const `_CharT *__s`, size_type __pos=0) const
- size_type `find_first_of` (const `_CharT *__s`, size_type __pos, size_type __n) const
- size_type `find_first_of` (const `_CharT *__s`, size_type __pos=0) const

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1043

- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`const basic_string &__str`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`const basic_string &__str`, `size_type __pos=0`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`const basic_string &__str`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_not_of` (`const basic_string &__str`, `size_type __pos=npos`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=npos`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=npos`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npos`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_of` (`const basic_string &__str`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_of` (`const basic_string &__str`, `size_type __pos=npos`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos=npos`) `const`
- `reference front` ()
- `const_reference front` () `const`
- `allocator_type get_allocator` () `const`
- `basic_string &insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `void insert` (`iterator __p`, `_InputIterator __beg`, `_InputIterator __end`)
- `void insert` (`iterator __p`, `initializer_list<_CharT> __l`)
- `basic_string &insert` (`size_type __pos1`, `const basic_string &__str`)
- `void insert` (`iterator __p`, `size_type __n`, `_CharT __c`)
- `basic_string &insert` (`size_type __pos`, `const _CharT *__s`, `size_type __n`)
- `basic_string &insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `basic_string &insert` (`size_type __pos1`, `const basic_string &__str`, `size_type __pos2`, `size_type __n`)

- `basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n)`
- `iterator insert (iterator __p, _CharT __c)`
- `basic_string & insert (size_type __pos1, const basic_string &__str)`
- `basic_string & insert (size_type __pos, const _CharT *__s)`
- `iterator insert (iterator __p, _CharT __c)`
- `void insert (iterator __p, size_type __n, _CharT __c)`
- `basic_string & insert (size_type __pos, const _CharT *__s)`
- `void insert (iterator __p, std::initializer_list< _CharT > __l)`
- `basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)`
- `template<typename _InputIterator > void insert (iterator __p, _InputIterator __first, _InputIterator __last)`
- `size_type length () const`
- `size_type max_size () const`
- `basic_string & operator+= (_CharT __c)`
- `basic_string & operator+= (std::initializer_list< _CharT > __l)`
- `basic_string & operator+= (const basic_string &__str)`
- `basic_string & operator+= (const _CharT *__s)`
- `basic_string & operator+= (_CharT __c)`
- `basic_string & operator+= (initializer_list< _CharT > __l)`
- `basic_string & operator+= (const basic_string &__str)`
- `basic_string & operator+= (const _CharT *__s)`
- `basic_string & operator= (std::initializer_list< _CharT > __l)`
- `basic_string & operator= (const basic_string &__str)`
- `basic_string & operator= (basic_string &&__str)`
- `basic_string & operator= (_CharT __c)`
- `basic_string & operator= (const _CharT *__s)`
- `const_reference operator[] (size_type __pos) const`
- `reference operator[] (size_type __pos)`
- `const_reference operator[] (size_type __pos) const`
- `reference operator[] (size_type __pos)`
- `void push_back (_CharT __c)`
- `void push_back (_CharT __c)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `reverse_iterator rend ()`

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1045

- `template<typename _InputIterator>`
`basic_string & replace (iterator __i1, iterator __i2, _InputIterator __j1, _InputIterator __j2)`
- `basic_string & replace (iterator __i1, iterator __i2, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)`
- `basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `basic_string & replace (size_type __pos, size_type __n, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)`
- `basic_string & replace (iterator __i1, iterator __i2, std::initializer_list< _CharT > __l)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __k1, const _CharT * __k2)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s)`
- `basic_string & replace (iterator __i1, iterator __i2, _CharT * __k1, _CharT * __k2)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`
- `basic_string & replace (iterator __i1, iterator __i2, iterator __k1, iterator __k2)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s)`
- `basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2)`
- `basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `void reserve (size_type __res_arg=0)`
- `void resize (size_type __n)`

- void [resize](#) (size_type __n, _CharT __c)
- void **resize** (size_type __n, _CharT __c)
- void **resize** (size_type __n)
- size_type [rfind](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **rfind** (const _CharT *__s, size_type __pos=[_Base::npos](#)) const
- size_type [rfind](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type **rfind** (const [basic_string](#) &__str, size_type __pos=[_Base::npos](#)) const
- size_type **rfind** (_CharT __c, size_type __pos=[_Base::npos](#)) const
- size_type [rfind](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [rfind](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type **rfind** (const _CharT *__s, size_type __pos, size_type __n) const
- void [shrink_to_fit](#) ()
- size_type [size](#) () const
- [basic_string](#) [substr](#) (size_type __pos=0, size_type __n=[npos](#)) const
- [basic_string](#) **substr** (size_type __pos=0, size_type __n=[_Base::npos](#)) const
- void **swap** ([basic_string](#)< _CharT, _Traits, _Allocator > &__x)
- void [swap](#) ([basic_string](#) &__s)

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Static Public Attributes

- static const size_type [npos](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) &__x)

5.89.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>,
typename _Allocator = std::allocator<_CharT>> class __gnu_debug::basic_
string<_CharT, _Traits, _Allocator>
```

Class `std::basic_string` with safety/checking/debug instrumentation.

Definition at line 42 of file `debug/string`.

5.89.2 Member Function Documentation

5.89.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.89.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.89.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_
base::_M_get_mutex () throw ()` [protected,
inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`,
and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.89.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.89.2.5 `void __gnu_debug::_Safe_sequence< basic_string< _CharT,
_Traits, _Allocator > >::_M_invalidate_if (_Predicate __pred)
[inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.89.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ()
[protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.89.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (
_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.89.2.8 `void __gnu_debug::_Safe_sequence< basic_string< _CharT,
_Traits, _Allocator > >::_M_transfer_iter (const _Safe_iterator<
_Iterator, basic_string< _CharT, _Traits, _Allocator > > & __x)
[inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.89.2.9 `basic_string& std::basic_string< _CharT, _Traits, _Allocator
>::append (const basic_string< _CharT, _Traits, _Allocator > &
__str) [inherited]`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1049

5.89.2.10 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n) [inherited]`

Append a substring.

Parameters

- str* The string to append.
- pos* Index of the first character of *str* to append.
- n* The number of characters to append.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function appends *n* characters from *str* starting at *pos* to this string. If *n* is larger than the number of available characters in *str*, the remainder of *str* is appended.

5.89.2.11 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const _CharT* __s, size_type __n) [inherited]`

Append a C substring.

Parameters

- s* The C string to append.
- n* The number of characters to append.

Returns

Reference to this string.

5.89.2.12 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const _CharT* __s) [inline, inherited]`

Append a C string.

Parameters

- s* The C string to append.

Returns

Reference to this string.

Definition at line 988 of file `basic_string.h`.

5.89.2.13 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (size_type __n, _CharT __c) [inherited]`

Append multiple characters.

Parameters

n The number of characters to append.

c The character to use.

Returns

Reference to this string.

Appends *n* copies of *c* to this string.

5.89.2.14 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (initializer_list<_CharT> __l) [inline, inherited]`

Append an `initializer_list` of characters.

Parameters

l The `initializer_list` of characters to append.

Returns

Reference to this string.

Definition at line 1012 of file `basic_string.h`.

5.89.2.15 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (_InputIterator __first, _InputIterator __last) [inline, inherited]`

Append a range of characters.

Parameters

first Iterator referencing the first character to append.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1051

last Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 1026 of file `basic_string.h`.

5.89.2.16 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const basic_string<_CharT, _Traits, _Allocator> & __str) [inherited]`

Set value to contents of another string.

Parameters

str Source string to use.

Returns

Reference to this string.

5.89.2.17 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (basic_string<_CharT, _Traits, _Allocator> && __str) [inline, inherited]`

Set value to contents of another string.

Parameters

str Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1061 of file `basic_string.h`.

5.89.2.18 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n) [inline, inherited]`

Set value to a substring of a string.

Parameters

str The string to use.
pos Index of the first character of *str*.
n Number of characters to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1081 of file `basic_string.h`.

5.89.2.19 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const _CharT* __s, size_type __n) [inherited]`

Set value to a C substring.

Parameters

s The C string to use.
n Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

5.89.2.20 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const _CharT* __s) [inline, inherited]`

Set value to contents of a C string.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1053

Parameters

s The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1109 of file `basic_string.h`.

5.89.2.21 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (size_type __n, _CharT __c) [inline, inherited]`

Set value to multiple characters.

Parameters

n Length of the resulting string.

c The character to use.

Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1125 of file `basic_string.h`.

5.89.2.22 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (_InputIterator __first, _InputIterator __last) [inline, inherited]`

Set value to a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 1138 of file `basic_string.h`.

5.89.2.23 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (initializer_list<_CharT> __l) [inline, inherited]`

Set value to an initializer_list of characters.

Parameters

l The initializer_list of characters to assign.

Returns

Reference to this string.

Definition at line 1148 of file basic_string.h.

5.89.2.24 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::at (size_type __n) const [inline, inherited]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 847 of file basic_string.h.

5.89.2.25 `reference std::basic_string<_CharT, _Traits, _Allocator>::at (size_type __n) [inline, inherited]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

[std::out_of_range](#) If n is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 900 of file `basic_string.h`.

5.89.2.26 `reference std::basic_string< _CharT, _Traits, _Allocator >::back () [inline, inherited]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 876 of file `basic_string.h`.

5.89.2.27 `const_reference std::basic_string< _CharT, _Traits, _Allocator >::back () const [inline, inherited]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 884 of file `basic_string.h`.

5.89.2.28 `iterator std::basic_string< _CharT, _Traits, _Allocator >::begin () [inline, inherited]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 591 of file `basic_string.h`.

5.89.2.29 `const_iterator std::basic_string< _CharT, _Traits, _Allocator >::begin () const [inline, inherited]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 602 of file `basic_string.h`.

5.89.2.30 `const _CharT* std::basic_string<_CharT, _Traits, _Allocator>::c_str () const [inline, inherited]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1757 of file basic_string.h.

5.89.2.31 `size_type std::basic_string<_CharT, _Traits, _Allocator>::capacity () const [inline, inherited]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 759 of file basic_string.h.

5.89.2.32 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::cbegin () const [inline, inherited]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 666 of file basic_string.h.

5.89.2.33 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::cend () const [inline, inherited]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 674 of file basic_string.h.

5.89.2.34 `void std::basic_string<_CharT, _Traits, _Allocator>::clear () [inline, inherited]`

Erases the string, making it empty.

Definition at line 786 of file basic_string.h.

5.89.2.35 `int std::basic_string<_CharT, _Traits, _Allocator>::compare (const _CharT * __s) const [inherited]`

Compare to a C string.

Parameters

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of *size()* and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.89.2.36 `int std::basic_string< _CharT, _Traits, _Allocator >::compare (const basic_string< _CharT, _Traits, _Allocator > & __str) const [inline, inherited]`

Compare to a string.

Parameters

str String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *str*, 0 if their values are equivalent, or > 0 if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of *size()* and *str.size()*. The function then compares the two strings by calling `traits::compare(data(), str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2164 of file `basic_string.h`.

5.89.2.37 `int std::basic_string< _CharT, _Traits, _Allocator >::compare (size_type __pos, size_type __n, const basic_string< _CharT, _Traits, _Allocator > & __str) const [inherited]`

Compare substring to a string.

Parameters

pos Index of first character of substring.

n Number of characters in substring.

str String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *str.size()*. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.89.2.38 `int std::basic_string<_CharT, _Traits, _Allocator >::compare (size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator > & __str, size_type __pos2, size_type __n2) const [inherited]`

Compare substring to a substring.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

str String to compare against.

pos2 Index of first character of substring of *str*.

n2 Number of characters in substring of *str*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form the substring of *str* from the *n2* characters starting at *pos2*. Returns an integer < 0 if this substring is ordered before the substring of *str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.89.2.39 `int std::basic_string< _CharT, _Traits, _Allocator >::compare (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) const` **[inherited]**

Compare substring against a character array.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

s character array to compare against.

n2 Number of characters of *s*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer < 0 if this substring is ordered before the string from *s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *s*. Determines the effective length *r1en* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling `traits::compare(substring.data(),s,r1en)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, `'\0'` has no special meaning.

5.89.2.40 `int std::basic_string< _CharT, _Traits, _Allocator >::compare (size_type __pos, size_type __n1, const _CharT * __s) const` **[inherited]**

Compare substring to a C string.

Parameters

pos Index of first character of substring.

n1 Number of characters in substring.

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or

> 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.89.2.41 `size_type std::basic_string<_CharT,_Traits,_Allocator>::copy`
`(_CharT* __s, size_type __n, size_type __pos = 0) const`
[inherited]

Copy substring into C string.

Parameters

- s* C string to copy value into.
- n* Number of characters to copy.
- pos* Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

[*std::out_of_range*](#) If *pos* > `size()`.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than `size()`, `out_of_range` is thrown.

5.89.2.42 `const_reverse_iterator std::basic_string<_CharT,_Traits,_Allocator>::crbegin() const` **[inline, inherited]**

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 683 of file `basic_string.h`.

5.89.2.43 `const_reverse_iterator std::basic_string<_CharT,_Traits,_Allocator>::crend() const` **[inline, inherited]**

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 692 of file `basic_string.h`.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1061

5.89.2.44 `const _CharT* std::basic_string<_CharT, _Traits, _Allocator>::data () const [inline, inherited]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1767 of file `basic_string.h`.

5.89.2.45 `bool std::basic_string<_CharT, _Traits, _Allocator>::empty () const [inline, inherited]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 794 of file `basic_string.h`.

5.89.2.46 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::end () const [inline, inherited]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 621 of file `basic_string.h`.

5.89.2.47 `iterator std::basic_string<_CharT, _Traits, _Allocator>::end () [inline, inherited]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 610 of file `basic_string.h`.

5.89.2.48 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::erase (size_type __pos = 0, size_type __n = npos) [inline, inherited]`

Remove characters.

Parameters

pos Index of first character to remove (default 0).

n Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are $< n$ characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, *out_of_range* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1338 of file *basic_string.h*.

5.89.2.49 `iterator std::basic_string<_CharT, _Traits, _Allocator >::erase (iterator __position) [inline, inherited]`

Remove one character.

Parameters

position Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1354 of file *basic_string.h*.

5.89.2.50 `iterator std::basic_string<_CharT, _Traits, _Allocator >::erase (iterator __first, iterator __last) [inherited]`

Remove a range of characters.

Parameters

first Iterator referencing the first character to remove.

last Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [*first*,*last*) from this string. The value of the string doesn't change if an error is thrown.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1063

5.89.2.51 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (const _CharT * __s, size_type __pos, size_type __n) const [inherited]`

Find position of a C substring.

Parameters

s C string to locate.

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns `npos`.

5.89.2.52 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0) const [inline, inherited]`

Find position of a string.

Parameters

str String to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1802 of file `basic_string.h`.

5.89.2.53 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (_CharT __c, size_type __pos = 0) const [inherited]`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.54 `size_type std::basic_string<_CharT, _Traits, _Allocator >::find
(const _CharT * __s, size_type __pos = 0) const [inline,
inherited]`

Find position of a C string.

Parameters

s C string to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1816 of file `basic_string.h`.

5.89.2.55 `size_type std::basic_string<_CharT, _Traits, _Allocator
>::find_first_not_of (const basic_string<_CharT, _Traits,
_Allocator > & __str, size_type __pos = 0) const [inline,
inherited]`

Find position of a character not in string.

Parameters

str String containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2026 of file `basic_string.h`.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1065

5.89.2.56 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of(const _CharT * __s, size_type __pos, size_type __n) const` `[inherited]`

Find position of a character not in C substring.

Parameters

- s* C string containing characters to avoid.
- pos* Index of character to search from.
- n* Number of characters from *s* to consider.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.57 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of(_CharT __c, size_type __pos = 0) const` `[inherited]`

Find position of a different character.

Parameters

- c* Character to avoid.
- pos* Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.58 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of(const _CharT * __s, size_type __pos = 0) const` `[inline, inherited]`

Find position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2055 of file basic_string.h.

5.89.2.59 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_first_of (_CharT __c, size_type __pos = 0) const
[inline, inherited]`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns npos.

Note: equivalent to find(*c*, *pos*).

Definition at line 1951 of file basic_string.h.

5.89.2.60 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_first_of (const _CharT * __s, size_type __pos = 0) const
[inline, inherited]`

Find position of a character of C string.

Parameters

s String containing characters to locate.

pos Index of character to search from (default 0).

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1067

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1932 of file `basic_string.h`.

```
5.89.2.61 size_type std::basic_string<_CharT, _Traits, _Allocator  
          >::find_first_of( const basic_string<_CharT, _Traits, _Allocator>  
          & __str, size_type __pos = 0 ) const [inline, inherited]
```

Find position of a character of string.

Parameters

str String containing characters to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1904 of file `basic_string.h`.

```
5.89.2.62 size_type std::basic_string<_CharT, _Traits, _Allocator  
          >::find_first_of( const _CharT * __s, size_type __pos, size_type  
          __n ) const [inherited]
```

Find position of a character of C substring.

Parameters

s String containing characters to locate.

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.63 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_not_of (const _CharT * __s, size_type __pos,
size_type __n) const [inherited]`

Find last position of a character not in C substring.

Parameters

- s* C string containing characters to avoid.
- pos* Index of character to search back from.
- n* Number of characters from *s* to consider.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.64 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_not_of (_CharT __c, size_type __pos = npo
s) const [inherited]`

Find last position of a different character.

Parameters

- c* Character to avoid.
- pos* Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.65 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_not_of (const _CharT * __s, size_type __pos = npo
s) const [inline, inherited]`

Find last position of a character not in C string.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1069

Parameters

- s* C string containing characters to avoid.
- pos* Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2114 of file `basic_string.h`.

5.89.2.66 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos) const` `[inline, inherited]`

Find last position of a character not in string.

Parameters

- str* String containing characters to avoid.
- pos* Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2085 of file `basic_string.h`.

5.89.2.67 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos) const` `[inline, inherited]`

Find last position of a character of string.

Parameters

- str* String containing characters to locate.
- pos* Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1965 of file `basic_string.h`.

```
5.89.2.68  size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_of ( const _CharT * __s, size_type __pos = npos )
const [inline, inherited]
```

Find last position of a character of C string.

Parameters

s C string containing characters to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1993 of file `basic_string.h`.

```
5.89.2.69  size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_of ( _CharT __c, size_type __pos = npos ) const
[inline, inherited]
```

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 2012 of file `basic_string.h`.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1071

5.89.2.70 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (const _CharT * __s, size_type __pos, size_type __n) const` `[inherited]`

Find last position of a character of C substring.

Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.71 `reference std::basic_string<_CharT, _Traits, _Allocator>::front ()` `[inline, inherited]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 860 of file `basic_string.h`.

5.89.2.72 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::front () const` `[inline, inherited]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 868 of file `basic_string.h`.

5.89.2.73 `allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator () const` `[inline, inherited]`

Return copy of allocator used to construct this string.

Definition at line 1774 of file `basic_string.h`.

5.89.2.74 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, size_type __n, _CharT __c)` `[inline, inherited]`

Insert multiple characters.

Parameters

- p* Iterator referencing location in string to insert at.
- n* Number of characters to insert
- c* The character to insert.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts *n* copies of character *c* starting at the position referenced by iterator *p*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1165 of file `basic_string.h`.

5.89.2.75 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n)`
[inline, inherited]

Insert a substring.

Parameters

- pos1* Iterator referencing location in string to insert at.
- str* The string to insert.
- pos2* Start of characters in *str* to insert.
- n* Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

std::out_of_range If *pos1* > `size()` or *pos2* > *str.size()*.

Starting at *pos1*, insert *n* character of *str* beginning with *pos2*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos1* is beyond the end of this string or *pos2* is beyond the end of *str*, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file `basic_string.h`.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1073

5.89.2.76 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, initializer_list<_CharT> __l) [inline, inherited]`

Insert an `initializer_list` of characters.

Parameters

- p* Iterator referencing location in string to insert at.
- l* The `initializer_list` of characters to insert.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Definition at line 1192 of file `basic_string.h`.

5.89.2.77 `iterator std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, _CharT __c) [inline, inherited]`

Insert one character.

Parameters

- p* Iterator referencing position in string to insert at.
- c* The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts character *c* at position referenced by *p*. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1314 of file `basic_string.h`.

5.89.2.78 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos, const basic_string<_CharT, _Traits, _Allocator> & __str) [inline, inherited]`

Insert value of a string.

Parameters

posI Iterator referencing location in string to insert at.

str The string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts value of *str* starting at *posI*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1211 of file `basic_string.h`.

5.89.2.79 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos, const _CharT* __s) [inline, inherited]`

Insert a C string.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1274 of file `basic_string.h`.

5.89.2.80 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos, size_type __n, _CharT __c)`
[`inline`, `inherited`]

Insert multiple characters.

Parameters

- pos* Index in string to insert at.
- n* Number of characters to insert
- c* The character to insert.

Returns

Reference to this string.

Exceptions

- [*std::length_error*](#) If new length exceeds `max_size()`.
- [*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts *n* copies of character *c* starting at index *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* > `length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1297 of file `basic_string.h`.

5.89.2.81 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos, const _CharT* __s, size_type __n)`
[`inherited`]

Insert a C substring.

Parameters

- pos* Iterator referencing location in string to insert at.
- s* The C string to insert.
- n* The number of characters to insert.

Returns

Reference to this string.

Exceptions

- [*std::length_error*](#) If new length exceeds `max_size()`.

std::out_of_range If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

5.89.2.82 `void std::basic_string<_CharT, _Traits, _Allocator >::insert
(iterator __p, _InputIterator __beg, _InputIterator __end)
[inline, inherited]`

Insert a range of characters.

Parameters

p Iterator referencing location in string to insert at.

beg Start of range.

end End of range.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1181 of file `basic_string.h`.

5.89.2.83 `size_type std::basic_string<_CharT, _Traits, _Allocator >::length () const [inline, inherited]`

Returns the number of characters in the string, not including any `///` null-termination.

Definition at line 707 of file `basic_string.h`.

5.89.2.84 `size_type std::basic_string<_CharT, _Traits, _Allocator >::max_size () const [inline, inherited]`

Returns the `size()` of the largest possible string.

Definition at line 712 of file `basic_string.h`.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1077

5.89.2.85 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=(const basic_string<_CharT, _Traits, _Allocator> & __str) [inline, inherited]`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

Definition at line 915 of file `basic_string.h`.

5.89.2.86 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=(const _CharT * __s) [inline, inherited]`

Append a C string.

Parameters

s The C string to append.

Returns

Reference to this string.

Definition at line 924 of file `basic_string.h`.

5.89.2.87 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=(_CharT __c) [inline, inherited]`

Append a character.

Parameters

c The character to append.

Returns

Reference to this string.

Definition at line 933 of file `basic_string.h`.

5.89.2.88 `basic_string& std::basic_string< _CharT, _Traits, _Allocator
>::operator+=(initializer_list< _CharT > __l) [inline,
inherited]`

Append an initializer_list of characters.

Parameters

l The initializer_list of characters to be appended.

Returns

Reference to this string.

Definition at line 946 of file basic_string.h.

5.89.2.89 `const_reference std::basic_string< _CharT, _Traits, _Allocator
>::operator[] (size_type __pos) const [inline, inherited]`

Subscript access to the data contained in the string.

Parameters

pos The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 809 of file basic_string.h.

5.89.2.90 `reference std::basic_string< _CharT, _Traits, _Allocator
>::operator[] (size_type __pos) [inline, inherited]`

Subscript access to the data contained in the string.

Parameters

pos The index of the character to access.

Returns

Read/write reference to the character.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1079

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 826 of file `basic_string.h`.

5.89.2.91 `void std::basic_string<_CharT, _Traits, _Allocator>::push_back (`
`_CharT __c) [inline, inherited]`

Append a single character.

Parameters

c Character to append.

Definition at line 1034 of file `basic_string.h`.

5.89.2.92 `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rbegin () const [inline, inherited]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 639 of file `basic_string.h`.

5.89.2.93 `reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rbegin () [inline, inherited]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 630 of file `basic_string.h`.

5.89.2.94 `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rend () const [inline, inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 657 of file `basic_string.h`.

5.89.2.95 `reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rend()` [**inline, inherited**]

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 648 of file `basic_string.h`.

5.89.2.96 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace(iterator __i1, iterator __i2, const _CharT* __s)` [**inline, inherited**]

Replace range of characters with C string.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file `basic_string.h`.

5.89.2.97 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace(iterator __i1, iterator __i2, const _CharT* __s, size_type __n)` [**inline, inherited**]

Replace range of characters with C substring.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1081

n Number of characters from *s* to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1517 of file `basic_string.h`.

5.89.2.98 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2) [inline, inherited]`

Replace range of characters with range.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

k1 Iterator referencing start of range to insert.

k2 Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1582 of file `basic_string.h`.

5.89.2.99 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, size_type __n, _CharT __c) [inline, inherited]`

Replace range of characters with multiple characters.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- n* Number of characters to insert.
- c* Character to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1559 of file `basic_string.h`.

5.89.2.100 `basic_string& std::basic_string<_CharT,_Traits,_Allocator>::replace (size_type __pos, size_type __n, const basic_string<_CharT,_Traits,_Allocator> & __str) [inline, inherited]`

Replace characters with value from another string.

Parameters

- pos* Index of first character to replace.
- n* Number of characters to be replaced.
- str* String to insert.

Returns

Reference to this string.

Exceptions

- [*std::out_of_range*](#) If *pos* is beyond the end of this string.
- [*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1393 of file `basic_string.h`.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1083

5.89.2.101 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n2) [inline, inherited]`

Replace characters with value from another string.

Parameters

pos1 Index of first character to replace.
n1 Number of characters to be replaced.
str String to insert.
pos2 Index of first character of str to use.
n2 Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos1* > size() or *pos2* > str.size().
std::length_error If new length exceeds max_size().

Removes the characters in the range [*pos1*, *pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, *out_of_range* is thrown. If the length of the result exceeds max_size(), *length_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1415 of file basic_string.h.

5.89.2.102 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, const basic_string<_CharT, _Traits, _Allocator> & __str) [inline, inherited]`

Replace range of characters with string.

Parameters

i1 Iterator referencing start of range to replace.
i2 Iterator referencing end of range to replace.
str String value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1499 of file `basic_string.h`.

5.89.2.103 `basic_string& std::basic_string<_CharT,_Traits,_Allocator>::replace(size_type __pos, size_type __n1, const _CharT* __s) [inline, inherited]`

Replace characters with value of a C string.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `pos > size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the characters of *s* are inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1458 of file `basic_string.h`.

5.89.2.104 `basic_string& std::basic_string<_CharT,_Traits,_Allocator>::replace(size_type __pos, size_type __n1, size_type __n2, _CharT __c) [inline, inherited]`

Replace characters with multiple characters.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1085

Parameters

- pos* Index of first character to replace.
- n1* Number of characters to be replaced.
- n2* Number of characters to insert.
- c* Character to insert.

Returns

Reference to this string.

Exceptions

- [*std::out_of_range*](#) If *pos* > `size()`.
- [*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1481 of file `basic_string.h`.

5.89.2.105 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, initializer_list<_CharT> __l) [inline, inherited]`

Replace range of characters with `initializer_list`.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- l* The `initializer_list` of characters to insert.

Returns

Reference to this string.

Exceptions

- [*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1, i2)`. In place, characters in the range `[k1, k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1650 of file `basic_string.h`.

5.89.2.106 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) [inherited]`

Replace characters with value of a C substring.

Parameters

pos Index of first character to replace.
n1 Number of characters to be replaced.
s C string to insert.
n2 Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos1* > size().
std::length_error If new length exceeds max_size().

Removes the characters in the range [pos,pos + n1) from this string. In place, the first *n2* characters of *s* are inserted, or all of *s* if *n2* is too large. If *pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

5.89.2.107 `void std::basic_string<_CharT, _Traits, _Allocator>::reserve (size_type __res_arg = 0) [inherited]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

res_arg Number of characters required.

Exceptions

std::length_error If *res_arg* exceeds max_size().

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

5.89.2.108 `void std::basic_string< _CharT, _Traits, _Allocator >::resize (size_type __n, _CharT __c) [inherited]`

Resizes the string to the specified number of characters.

Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

5.89.2.109 `void std::basic_string< _CharT, _Traits, _Allocator >::resize (size_type __n) [inline, inherited]`

Resizes the string to the specified number of characters.

Parameters

- n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 739 of file basic_string.h.

5.89.2.110 `size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (const basic_string< _CharT, _Traits, _Allocator > & __str, size_type __pos = npos) const [inline, inherited]`

Find last position of a string.

Parameters

- str* String to locate.
- pos* Index of character to search back from (default end).

Returns

- Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1846 of file `basic_string.h`.

5.89.2.111 `size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (_CharT __c, size_type __pos = npos) const [inherited]`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.89.2.112 `size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (const _CharT * __s, size_type __pos = npos) const [inline, inherited]`

Find last position of a C string.

Parameters

s C string to locate.

pos Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1874 of file `basic_string.h`.

5.89.2.113 `size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (const _CharT * __s, size_type __pos, size_type __n) const [inherited]`

Find last position of a C substring.

5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1089

Parameters

- s* C string to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

5.89.2.114 `void std::basic_string<_CharT, _Traits, _Allocator>::shrink_to_fit () [inline, inherited]`

A non-binding request to reduce capacity() to size().

Definition at line 745 of file `basic_string.h`.

5.89.2.115 `size_type std::basic_string<_CharT, _Traits, _Allocator>::size () const [inline, inherited]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 701 of file `basic_string.h`.

5.89.2.116 `basic_string std::basic_string<_CharT, _Traits, _Allocator>::substr (size_type __pos = 0, size_type __n = npos) const [inline, inherited]`

Get a substring.

Parameters

- pos* Index of first character (default 0).
- n* Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

[*std::out_of_range*](#) If *pos* > size().

Construct and return a new string using the n characters starting at pos . If the string is too short, use the remainder of the characters. If pos is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2146 of file `basic_string.h`.

5.89.2.117 `void std::basic_string<_CharT, _Traits, _Allocator>::swap
(basic_string<_CharT, _Traits, _Allocator> & __s)
[inherited]`

Swap contents with another string.

Parameters

s String to swap with.

Exchanges the contents of this string with that of s in constant time.

5.89.3 Member Data Documentation

5.89.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_
const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.89.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_
iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.89.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version
[mutable, inherited]`

The container version number. This number may never be 0.

5.90 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference 1091

Definition at line 169 of file `safe_base.h`.

5.89.3.4 `const size_type std::basic_string<_CharT, _Traits, _Allocator>::npos` `[static, inherited]`

Value returned by various member functions when they fail.

Definition at line 271 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

5.90 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

General reduction, using a binary operator.

Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp` &__b)
- `template<typename _Result, typename _Addend> _Result operator()` (`const _Result` &__x, `const _Addend` &__y)

Public Attributes

- `_BinOp` & `__binop`

5.90.1 Detailed Description

`template<typename _BinOp> struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>`

General reduction, using a binary operator.

Definition at line 335 of file `for_each_selectors.h`.

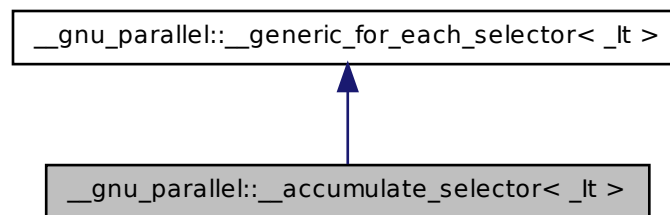
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.91 `__gnu_parallel::__accumulate_selector< _It >` Struct Template Reference

[std::accumulate\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__accumulate_selector< _It >`:



Public Member Functions

- `template<typename _Op >`
[std::iterator_traits< _It >::value_type](#) [operator\(\)](#) (`_Op __o`, `_It __i`)

Public Attributes

- `_It` [_M_finish_iterator](#)

5.91.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__accumulate_selector< _It >`

[std::accumulate\(\)](#) selector.

Definition at line 208 of file `for_each_selectors.h`.

5.91.2 Member Function Documentation

5.91.2.1 `template<typename _It> template<typename _Op> std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator() (_Op __o, _It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator (unused).
- `__i` iterator referencing object.

Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

5.91.3 Member Data Documentation

5.91.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

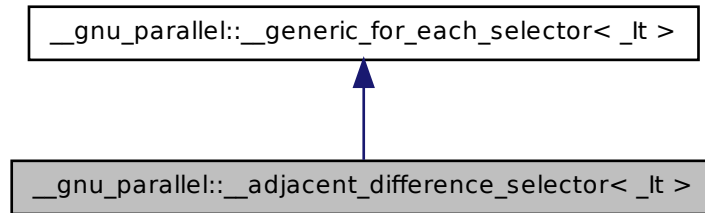
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.92 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Selector that returns the difference between two adjacent `__elements`.

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.92.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__adjacent_difference_selector<_It>`

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

5.92.2 Member Data Documentation

5.92.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file for_each_selectors.h.

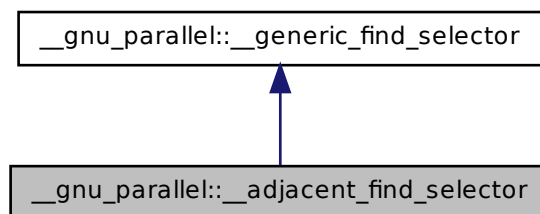
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.93 __gnu_parallel::__adjacent_find_selector Struct Reference

Test predicate on two adjacent elements.

Inheritance diagram for __gnu_parallel::__adjacent_find_selector:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-`
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

5.93.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file find_selectors.h.

5.93.2 Member Function Documentation

5.93.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__adjacent_find_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

`__begin1` Begin iterator of first sequence.
`__end1` End iterator of first sequence.
`__begin2` Begin iterator of second sequence.
`__pred` Find predicate.

Definition at line 105 of file `find_selectors.h`.

References `std::make_pair()`.

5.93.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__adjacent_find_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

`__i1` Iterator on first sequence.
`__i2` Iterator on second sequence (unused).
`__pred` Find predicate.

Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

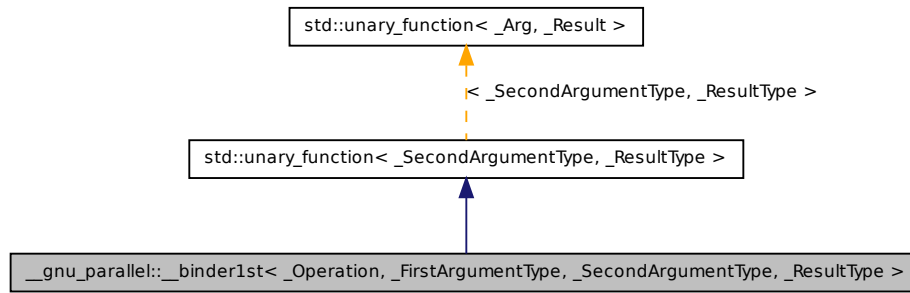
- [find_selectors.h](#)

5.94 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Similar to [std::binder1st](#), but giving the argument types explicitly.

5.94 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference 1097

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



Public Types

- typedef `_SecondArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- `__binder1st` (`const _Operation &__x`, `const _FirstArgumentType &__y`)
- `_ResultType operator()` (`_SecondArgumentType &__x`) `const`
- `_ResultType operator()` (`const _SecondArgumentType &__x`)

Protected Attributes

- `_Operation _M_op`
- `_FirstArgumentType _M_value`

5.94.1 Detailed Description

template<typename `_Operation`, typename `_FirstArgumentType`, typename `_SecondArgumentType`, typename `_ResultType`> class `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`

Similar to `std::binder1st`, but giving the argument types explicitly.

Definition at line 192 of file parallel/base.h.

5.94.2 Member Typedef Documentation

5.94.2.1 `typedef _SecondArgumentType std::unary_function<
_SecondArgumentType , _ResultType >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.94.2.2 `typedef _ResultType std::unary_function< _SecondArgumentType ,
_ResultType >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file stl_function.h.

The documentation for this class was generated from the following file:

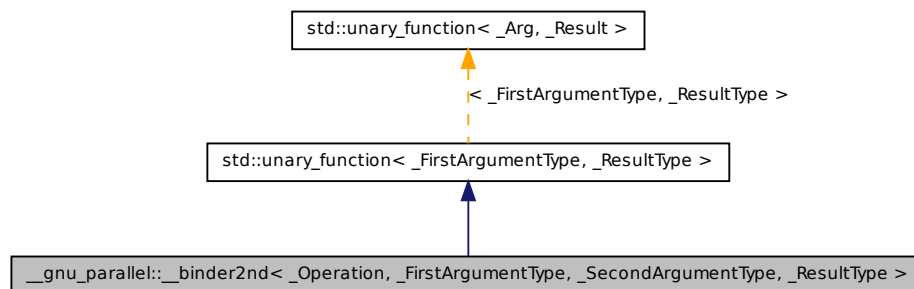
- [parallel/base.h](#)

5.95 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Similar to [std::binder2nd](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _`

`FirstArgumentType, _SecondArgumentType, _ResultType >:`



Public Types

- typedef `_FirstArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- `__binder2nd` (`const _Operation &__x, const _SecondArgumentType &__y`)
- `_ResultType operator()` (`_FirstArgumentType &__x`)
- `_ResultType operator()` (`const _FirstArgumentType &__x`) `const`

Protected Attributes

- `_Operation _M_op`
- `_SecondArgumentType _M_value`

5.95.1 Detailed Description

`template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType> class __gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`

Similar to [std::binder2nd](#), but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

5.95.2 Member Typedef Documentation

5.95.2.1 `typedef _FirstArgumentType std::unary_function<
_FirstArgumentType , _ResultType >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.95.2.2 `typedef _ResultType std::unary_function< _FirstArgumentType ,
_ResultType >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

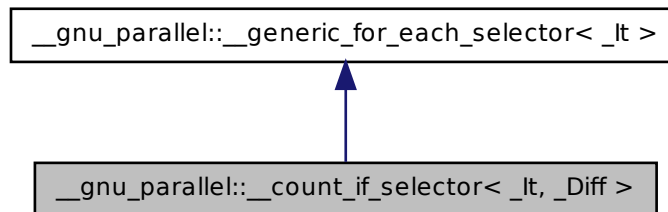
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.96 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference

`std::count_if()` selector.

Inheritance diagram for `__gnu_parallel::__count_if_selector< _It, _Diff >`:



Public Member Functions

- `template<typename _Op> _Diff operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.96.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_if_selector<_It, _Diff>`

`std::count_if()` selector.

Definition at line 194 of file `for_each_selectors.h`.

5.96.2 Member Function Documentation

5.96.2.1 `template<typename _It, typename _Diff> template<typename _Op> > _Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator() (_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

5.96.3 Member Data Documentation

5.96.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

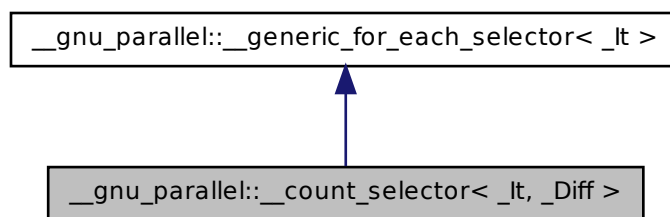
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.97 `__gnu_parallel::__count_selector< _It, _Diff >` Struct Template Reference

`std::count()` selector.

Inheritance diagram for `__gnu_parallel::__count_selector< _It, _Diff >`:



Public Member Functions

- `template<typename _ValueType >`
`_Diff operator\(\) (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.97 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

5.97.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_selector<_It, _Diff>`

`std::count()` selector.

Definition at line 180 of file `for_each_selectors.h`.

5.97.2 Member Function Documentation

5.97.2.1 `template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_selector<_It, _Diff>::operator() (_ValueType & __v, _It __i) [inline]`

Functor execution.

Parameters

`__v` Current value.

`__i` iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

5.97.3 Member Data Documentation

5.97.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

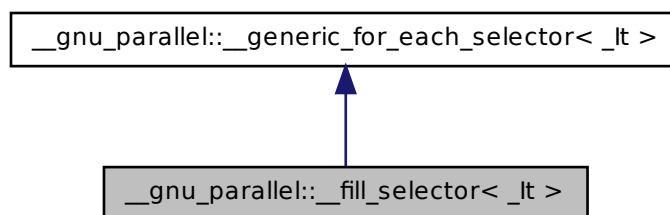
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.98 `__gnu_parallel::__fill_selector< _It >` Struct Template Reference

`std::fill()` selector.

Inheritance diagram for `__gnu_parallel::__fill_selector< _It >`:



Public Member Functions

- `template<typename _ValueType > bool operator() (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.98.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__fill_selector< _It >`

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

5.98.2 Member Function Documentation

5.98.2.1 `template<typename _It> template<typename _ValueType> bool
__gnu_parallel::__fill_selector<_It>::operator() (_ValueType &
__v, _It __i) [inline]`

Functor execution.

Parameters

`__v` Current value.

`__i` iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

5.98.3 Member Data Documentation

5.98.3.1 `template<typename _It> _It __gnu_parallel::__-
generic_for_each_selector<_It>::M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

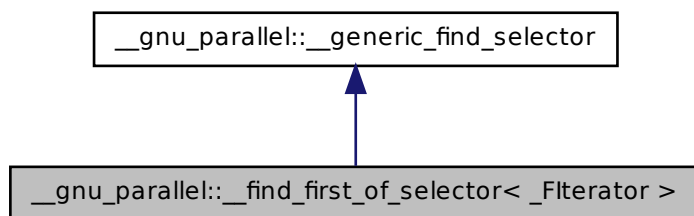
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.99 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Test predicate on several elements.

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



Public Member Functions

- **`__find_first_of_selector`** (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`std::pair<_RAIter1, _RAIter2> _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

Public Attributes

- `_FIterator _M_begin`
- `_FIterator _M_end`

5.99.1 Detailed Description

`template<typename _FIterator> struct __gnu_parallel::__find_first_of_selector<_FIterator>`

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

5.99.2 Member Function Documentation

5.99.2.1 `template<typename _FIterator > template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

- __begin1* Begin iterator of first sequence.
- __end1* End iterator of first sequence.
- __begin2* Begin iterator of second sequence.
- __pred* Find predicate.

Definition at line 186 of file `find_selectors.h`.

References `std::make_pair()`.

5.99.2.2 `template<typename _FIterator > template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__find_first_of_selector<_FIterator>::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

- __i1* Iterator on first sequence.
- __i2* Iterator on second sequence (unused).
- __pred* Find predicate.

Definition at line 169 of file `find_selectors.h`.

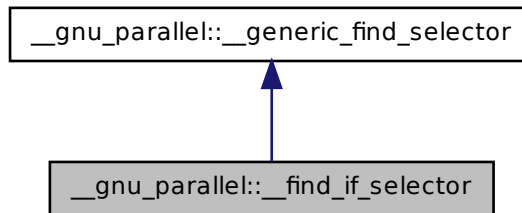
The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.100 `__gnu_parallel::__find_if_selector` Struct Reference

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-`
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

5.100.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

5.100.2 Member Function Documentation

5.100.2.1 `template<typename _RAIter1, typename _RAIter2`
`, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__find_if_selector::__M_sequential_algorithm (`
`__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __Pred`
`__pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

`__begin1` Begin iterator of first sequence.

5.101 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference 109

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence.

`__pred` Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

```
5.100.2.2  template<typename _RAIter1 , typename _RAIter2 , typename
            _Pred > bool __gnu_parallel::__find_if_selector::operator() (
            _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]
```

Test on one position.

Parameters

`__i1` _Iterator on first sequence.

`__i2` _Iterator on second sequence (unused).

`__pred` Find predicate.

Definition at line 60 of file `find_selectors.h`.

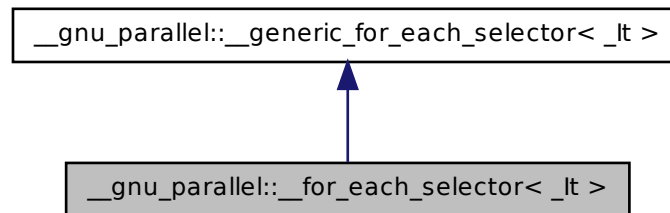
The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.101 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

`std::for_each()` selector.

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It` `_M_finish_iterator`

5.101.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__for_each_selector<_It>`

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

5.101.2 Member Function Documentation

5.101.2.1 `template<typename _It> template<typename _Op> bool`
`__gnu_parallel::__for_each_selector<_It>::operator() (_Op &`
`__o, _It __i) [inline]`

Functor execution.

5.102 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference 111

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

5.101.3 Member Data Documentation

5.101.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

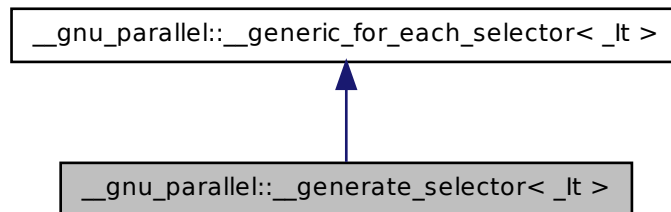
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.102 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

`std::generate()` selector.

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



Public Member Functions

- `template<typename _Op >`
`bool operator\(\) (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.102.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector< _It >`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

5.102.2 Member Function Documentation

5.102.2.1 `template<typename _It> template<typename _Op > bool`
`__gnu_parallel::__generate_selector< _It >::operator() (_Op &`
`__o, _It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

5.102.3 Member Data Documentation

5.102.3.1 `template<typename _It > _It __gnu_parallel::__`
`generic_for_each_selector< _It >::__M_finish_iterator`
`[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

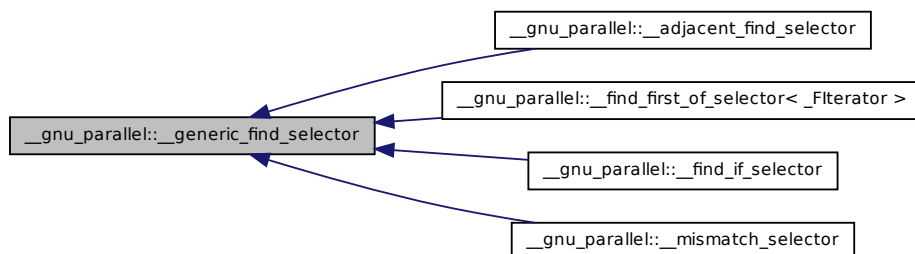
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.103 __gnu_parallel::__generic_find_selector Struct Reference

Base class of all [__gnu_parallel::__find_template](#) selectors.

Inheritance diagram for __gnu_parallel::__generic_find_selector:



5.103.1 Detailed Description

Base class of all [__gnu_parallel::__find_template](#) selectors.

Definition at line 43 of file `find_selectors.h`.

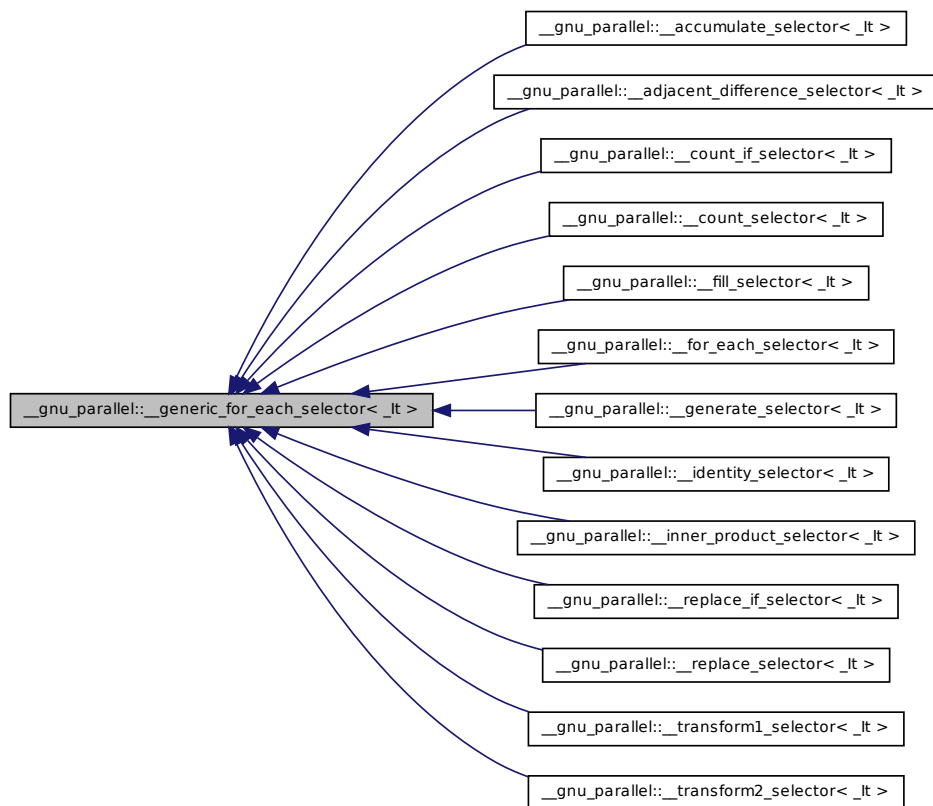
The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.104 __gnu_parallel::__generic_for_each_selector<_It> Struct Template Reference

Generic `__selector` for embarrassingly parallel functions.

Inheritance diagram for `__gnu_parallel::__generic_for_each_selector<_It>`:



Public Attributes

- `_It` [_M_finish_iterator](#)

5.104.1 Detailed Description

```
template<typename _It> struct __gnu_parallel::__generic_for_each_selector<
_It>
```

Generic `__selector` for embarrassingly parallel functions.

5.105 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

Definition at line 42 of file `for_each_selectors.h`.

5.104.2 Member Data Documentation

5.104.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

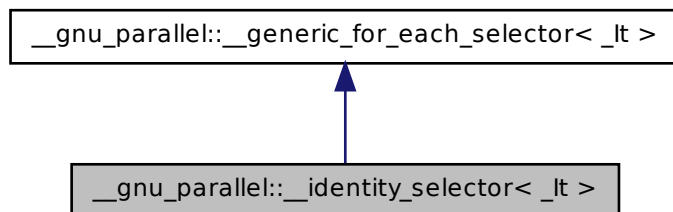
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.105 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

Selector that just returns the passed iterator.

Inheritance diagram for `__gnu_parallel::__identity_selector<_It>`:



Public Member Functions

- `template<typename _Op> _It operator() (_Op __o, _It __i)`

Public Attributes

- [_It __M_finish_iterator](#)

5.105.1 Detailed Description

template<typename _It> struct __gnu_parallel::__identity_selector< _It >

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

5.105.2 Member Function Documentation

**5.105.2.1 template<typename _It> template<typename _Op > _It
__gnu_parallel::__identity_selector< _It >::operator() (_Op __o,
_It __i) [inline]**

Functor execution.

Parameters

- `__o` Operator (unused).
- `__i` iterator referencing object.

Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

5.105.3 Member Data Documentation

**5.105.3.1 template<typename _It > _It __gnu_parallel::__-
generic_for_each_selector< _It >::__M_finish_iterator
[inherited]**

__Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

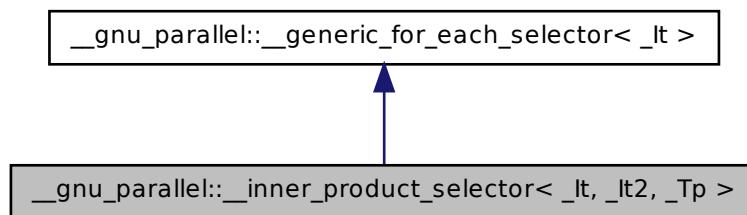
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.106 `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct` Template Reference

[std::inner_product\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`:



Public Member Functions

- [__inner_product_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op >`
`_Tp operator\(\) (_Op __mult, _It __current)`

Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

5.106.1 Detailed Description

`template<typename _It, typename _It2, typename _Tp> struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`

[std::inner_product\(\)](#) selector.

Definition at line 222 of file `for_each_selectors.h`.

5.106.2 Constructor & Destructor Documentation

5.106.2.1 `template<typename _It , typename _It2 , typename _Tp >
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp
>::__inner_product_selector (_It __b1, _It2 __b2) [inline,
explicit]`

Constructor.

Parameters

- b1* Begin iterator of first sequence.
- b2* Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

5.106.3 Member Function Documentation

5.106.3.1 `template<typename _It , typename _It2 , typename _Tp >
template<typename _Op > _Tp __gnu_parallel::__inner_product_
selector< _It, _It2, _Tp >::operator() (_Op __mult, _It __current)
[inline]`

Functor execution.

Parameters

- __mult* Multiplication functor.
- __current* iterator referencing object.

Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_` - iterator, and `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_` - iterator.

5.106.4 Member Data Documentation

5.106.4.1 `template<typename _It , typename _It2 , typename _Tp >
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp
>::__begin1_iterator`

Begin iterator of first sequence.

5.107 `__gnu_parallel::__max_element_reduct<_Compare, _It> Struct` 1119 Template Reference

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

5.106.4.2 `template<typename _It, typename _It2, typename _Tp>
_It2 __gnu_parallel::__inner_product_selector<_It, _It2, _Tp
>::__begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

5.106.4.3 `template<typename _It> _It __gnu_parallel::__-
generic_for_each_selector<_It>::__M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.107 `__gnu_parallel::__max_element_reduct<_Compare, _It> Struct` - Template Reference

Reduction for finding the maximum element, using a comparator.

Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

Public Attributes

- `_Compare & __comp`

5.107.1 Detailed Description

template<typename _Compare, typename _It> struct __gnu_parallel::__max_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.108 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference

Reduction for finding the maximum element, using a comparator.

Public Member Functions

- `__min_element_reduct (_Compare &__c)`
- `_It operator() (_It __x, _It __y)`

Public Attributes

- `_Compare & __comp`

5.108.1 Detailed Description

template<typename _Compare, typename _It> struct __gnu_parallel::__min_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.

Definition at line 307 of file `for_each_selectors.h`.

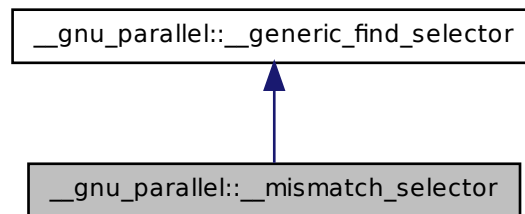
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.109 __gnu_parallel::__mismatch_selector Struct Reference

Test inverted predicate on a single element.

Inheritance diagram for __gnu_parallel::__mismatch_selector:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-`
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

5.109.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

5.109.2 Member Function Documentation

5.109.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

`__begin1` Begin iterator of first sequence.
`__end1` End iterator of first sequence.
`__begin2` Begin iterator of second sequence.
`__pred` Find predicate.

Definition at line 143 of file `find_selectors.h`.

5.109.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

`__i1` Iterator on first sequence.
`__i2` Iterator on second sequence (unused).
`__pred` Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.110 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 3-way merging with `__sentinels` turned off.

5.111 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference` 1123

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.110.1 Detailed Description

`template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned off. Note that 3-way merging is always stable!

Definition at line 748 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.111 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference`

Switch for 3-way merging with `__sentinels` turned on.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.111.1 Detailed Description

`template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned on. Note that 3-way merging is always stable!

Definition at line 768 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.112 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 4-way merging with `__sentinels` turned off.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.112.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__
multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _
RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned off. Note that 4-way merging is always stable!

Definition at line 791 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.113 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 4-way merging with `__sentinels` turned on.

5.114 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch<__sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare>` 1125

Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.113.1 Detailed Description

```
template<typename _RAIterIterator, typename _RAIter3, typename _-
DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_-
merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _-
DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on. Note that 4-way merging is always stable!

Definition at line 811 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.114 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare>` 1125

Struct Template Reference

Switch for k-way merging with `__sentinels` turned on.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

5.114.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterIterator, type-
name _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_
parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable,
_RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with __sentinels turned on.

Definition at line 833 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.115 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Switch for k-way merging with __sentinels turned off.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

5.115.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__-
multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _-
RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with __sentinels turned off.

Definition at line 868 of file multiway_merge.h.

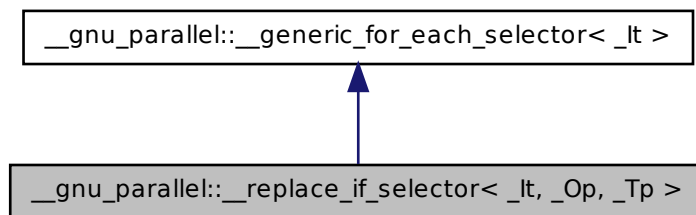
The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.116 `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



Public Member Functions

- `__replace_if_selector` (`const _Tp &__new_val`)
- `bool operator()` (`_Op &__o, _It __i`)

Public Attributes

- `const _Tp & __new_val`
- `_It __M_finish_iterator`

5.116.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp> struct __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>
```

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

5.116.2 Constructor & Destructor Documentation

5.116.2.1 `template<typename _It , typename _Op , typename _Tp
> __gnu_parallel::__replace_if_selector< _It, _Op, _Tp
>::__replace_if_selector(const _Tp & __new_val) [inline,
explicit]`

Constructor.

Parameters

`__new_val` Value to replace with.

Definition at line 164 of file `for_each_selectors.h`.

5.116.3 Member Function Documentation

5.116.3.1 `template<typename _It , typename _Op , typename _Tp > bool
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val`.

5.116.4 Member Data Documentation

5.116.4.1 `template<typename _It , typename _Op , typename _Tp > const
_Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp
>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator()()`.

5.117 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

5.116.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`
[`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

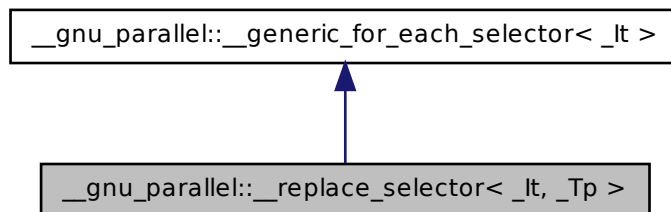
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.117 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



Public Member Functions

- `__replace_selector` (`const _Tp &__new_val`)
- `bool operator()` (`_Tp &__v, _It __i`)

Public Attributes

- `const _Tp &__new_val`
- `_It __M_finish_iterator`

5.117.1 Detailed Description

```
template<typename _It, typename _Tp> struct __gnu_parallel::__replace_
selector< _It, _Tp >
```

std::replace() selector.

Definition at line 132 of file for_each_selectors.h.

5.117.2 Constructor & Destructor Documentation

5.117.2.1 `template<typename _It, typename _Tp > __gnu_parallel::__replace_selector< _It, _Tp >::__replace_selector (const _Tp & __new_val) [inline, explicit]`

Constructor.

Parameters

`__new_val` Value to replace with.

Definition at line 140 of file for_each_selectors.h.

5.117.3 Member Function Documentation

5.117.3.1 `template<typename _It, typename _Tp > bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (_Tp & __v, _It __i) [inline]`

Functor execution.

Parameters

`__v` Current value.

`__i` iterator referencing object.

Definition at line 146 of file for_each_selectors.h.

References `__gnu_parallel::__replace_selector< _It, _Tp >::__new_val`.

5.117.4 Member Data Documentation

5.117.4.1 `template<typename _It, typename _Tp > const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val`

Value to replace with.

5.118 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp>::operator()`.

5.117.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

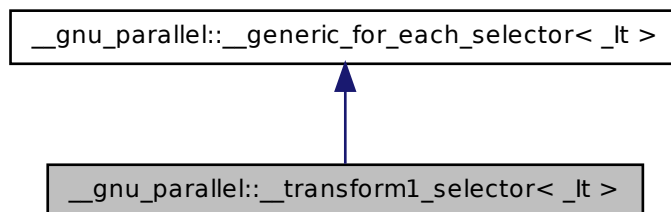
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.118 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

`std::transform()` `__selector`, one input sequence variant.

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

5.118.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform1_selector< _It >`

`std::transform()` __selector, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

5.118.2 Member Function Documentation

5.118.2.1 `template<typename _It> template<typename _Op > bool
__gnu_parallel::__transform1_selector< _It >::operator() (_Op &
__o, _It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

5.118.3 Member Data Documentation

5.118.3.1 `template<typename _It > _It __gnu_parallel::__
generic_for_each_selector< _It >::__M_finish_iterator
[inherited]`

__Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

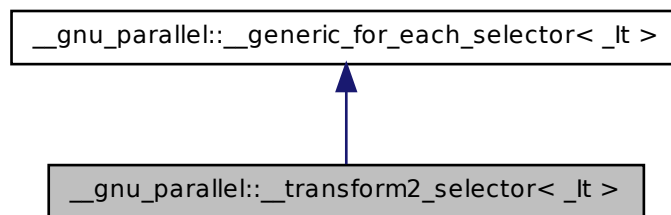
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.119 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference

`std::transform()` `__selector`, two input sequences variant.

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.119.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform2_selector<_It>`

`std::transform()` `__selector`, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

5.119.2 Member Function Documentation

5.119.2.1 `template<typename _It> template<typename _Op > bool
__gnu_parallel::__transform2_selector< _It >::operator() (_Op &
__o, _It __i) [inline]`

Functor execution.

Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

5.119.3 Member Data Documentation

5.119.3.1 `template<typename _It > _It __gnu_parallel::__
generic_for_each_selector< _It >::__M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

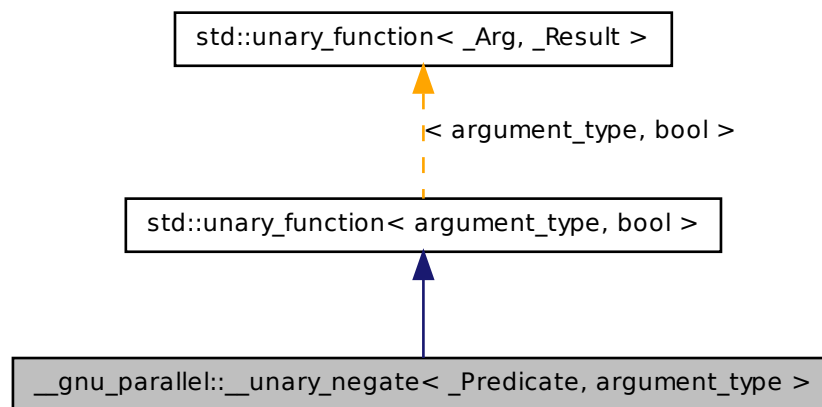
- [for_each_selectors.h](#)

5.120 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference

Similar to [std::unary_negate](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__unary_negate< _Predicate, argument_type`

>:



Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- `bool operator()` (`const argument_type &__x`)

Protected Attributes

- `_Predicate _M_pred`

5.120.1 Detailed Description

`template<typename _Predicate, typename argument_type> class __gnu_parallel::__unary_negate<_Predicate, argument_type>`

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file parallel/base.h.

5.120.2 Member Typedef Documentation

5.120.2.1 `typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.120.2.2 `typedef bool std::unary_function< argument_type , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file stl_function.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.121 `__gnu_parallel::__DRandomShufflingGlobalData< _RAIter >` Struct Template Reference

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator_traits< _RAIter > _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

Public Member Functions

- `_DRandomShufflingGlobalData (_RAIter &__source)`

Public Attributes

- `_ThreadIndex * _M_bin_proc`

- `_DifferenceType ** _M_dist`
- `int _M_num_bins`
- `int _M_num_bits`
- `_RAIter & _M_source`
- `_DifferenceType * _M_starts`
- `_ValueType ** _M_temporaries`

5.121.1 Detailed Description

template<typename _RAIter> struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>

Data known to every thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Definition at line 52 of file `random_shuffle.h`.

5.121.2 Constructor & Destructor Documentation

5.121.2.1 **template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__DRandomShufflingGlobalData (_RAIter & __source)**
[inline]

Constructor.

Definition at line 83 of file `random_shuffle.h`.

5.121.3 Member Data Documentation

5.121.3.1 **template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_bin_proc**

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#).

5.121.3.2 `template<typename _RAIter> _DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist`

Two-dimensional array to hold the thread-bin distribution.

Dimensions (`_M_num_threads + 1`) `__x` (`_M_num_bins + 1`).

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.121.3.3 `template<typename _RAIter> int __gnu_ parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins`

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.121.3.4 `template<typename _RAIter> int __gnu_ parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.121.3.5 `template<typename _RAIter> _RAIter& __gnu_ parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

5.122 `__gnu_parallel::__DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference 1139

5.121.3.6 `template<typename _RAIter> _DifferenceType*
__gnu_parallel::__DRandomShufflingGlobalData<_RAIter
>::__M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.121.3.7 `template<typename _RAIter> _ValueType**
__gnu_parallel::__DRandomShufflingGlobalData<_RAIter
>::__M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

5.122 `__gnu_parallel::__DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Public Attributes

- [_BinIndex __bins_end](#)
- [_BinIndex _M_bins_begin](#)
- [int _M_num_threads](#)
- [_DRandomShufflingGlobalData<_RAIter> * _M_sd](#)
- [uint32_t _M_seed](#)

5.122.1 Detailed Description

template<typename _RAIter, typename _RandomNumberGenerator> struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Definition at line 91 of file random_shuffle.h.

5.122.2 Member Data Documentation

5.122.2.1 template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end

End index for bins taken care of by this thread.

Definition at line 100 of file random_shuffle.h.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#).

5.122.2.2 template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_bins_begin

Begin index for bins taken care of by this thread.

Definition at line 97 of file random_shuffle.h.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#).

5.122.2.3 template<typename _RAIter, typename _RandomNumberGenerator> int __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_num_threads

Number of threads participating in total.

Definition at line 94 of file random_shuffle.h.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#), and [__gnu_parallel::__parallel_random_shuffle_drs_pu\(\)](#).

5.122.2.4 `template<typename _RAIter, typename
_RandomNumberGenerator> _DRandomShufflingGlobalData<_
_RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter,
_RandomNumberGenerator >::_M_sd`

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.122.2.5 `template<typename _RAIter, typename
_RandomNumberGenerator> uint32_t __gnu_parallel::_
DRSSorterPU< _RAIter, _RandomNumberGenerator
>::_M_seed`

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

5.123 __gnu_parallel::_DummyReduct Struct Reference

Reduction function doing nothing.

Public Member Functions

- `bool operator() (bool, bool) const`

5.123.1 Detailed Description

Reduction function doing nothing.

Definition at line 298 of file `for_each_selectors.h`.

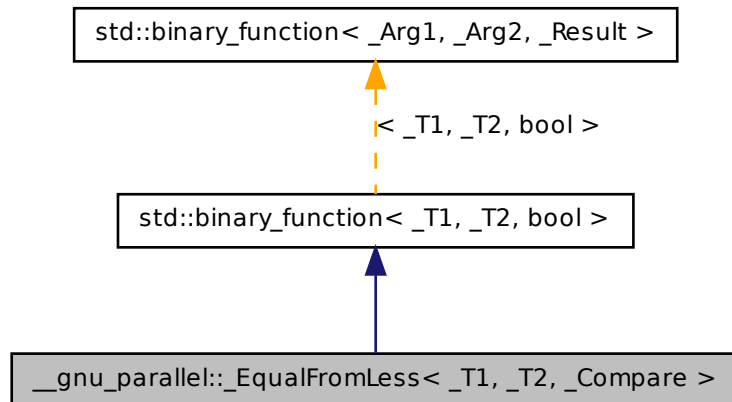
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.124 `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >` Class Template Reference

Constructs predicate for equality from strict weak ordering predicate.

Inheritance diagram for `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `_EqualFromLess` (`_Compare &__comp`)
- `bool operator()` (`const _T1 &__a, const _T2 &__b`)

5.124.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file `parallel/base.h`.

5.124.2 Member Typedef Documentation

5.124.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.124.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.124.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

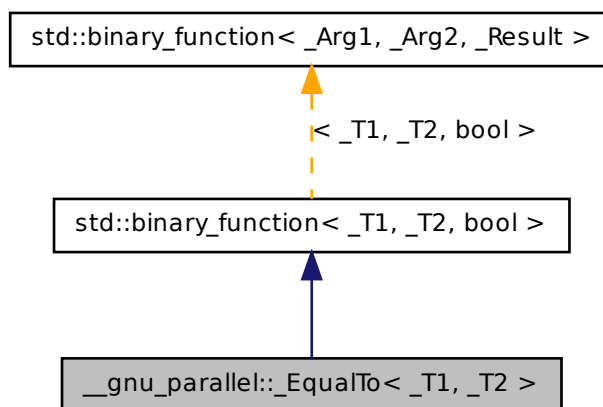
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.125 `__gnu_parallel::_EqualTo< _T1, _T2 >` Struct Template Reference

Similar to `std::equal_to`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_EqualTo<_T1, _T2>`:



Public Types

- `typedef _T1` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _T2` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

5.125.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_EqualTo<_T1, _T2>
```

Similar to [std::equal_to](#), but allows two different types.

Definition at line 244 of file `parallel/base.h`.

5.125.2 Member Typedef Documentation

5.125.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.125.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.125.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.126 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

Public Member Functions

- [_GuardedIterator](#) (_RAIter __begin, _RAIter __end, _Compare &__comp)
- [operator _RAIter](#) ()
- [std::iterator_traits< _RAIter >::value_type & operator*](#) ()
- [_GuardedIterator](#)< _RAIter, _Compare > & [operator++](#) ()

Friends

- `bool operator< (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`
- `bool operator<= (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`

5.126.1 Detailed Description

`template<typename _RAIter, typename _Compare> class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >`

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons. The implicit supremum comes with a performance cost.

Deriving from _RAIter is not possible since _RAIter need not be a class.

Definition at line 66 of file multiway_merge.h.

5.126.2 Constructor & Destructor Documentation

5.126.2.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::_GuardedIterator (_RAIter __begin, _RAIter __end, _Compare & __comp) [inline]`

Constructor. Sets iterator to beginning of sequence.

Parameters

`__begin` Begin iterator of sequence.

`__end` End iterator of sequence.

`__comp` Comparator provided for associated overloaded compare operators.

Definition at line 84 of file multiway_merge.h.

5.126.3 Member Function Documentation

5.126.3.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () [inline]`

Convert to wrapped iterator.

5.126 __gnu_parallel::_GuardedIterator<_RAIter, _Compare > Class Template Reference 1147

Returns

Wrapped iterator.

Definition at line 105 of file multiway_merge.h.

```
5.126.3.2 template<typename _RAIter , typename _Compare  
            > std::iterator_traits<_RAIter>::value_type&  
            __gnu_parallel::_GuardedIterator< _RAIter, _Compare  
            >::operator*( ) [inline]
```

Dereference operator.

Returns

Referenced element.

Definition at line 100 of file multiway_merge.h.

```
5.126.3.3 template<typename _RAIter , typename _Compare  
            > _GuardedIterator<_RAIter, _Compare>&  
            __gnu_parallel::_GuardedIterator< _RAIter, _Compare  
            >::operator++( ) [inline]
```

Pre-increment operator.

Returns

This.

Definition at line 91 of file multiway_merge.h.

5.126.4 Friends And Related Function Documentation

```
5.126.4.1 template<typename _RAIter , typename _Compare > bool  
            operator< ( _GuardedIterator< _RAIter, _Compare > & __bi1,  
            _GuardedIterator< _RAIter, _Compare > & __bi2 ) [friend]
```

Compare two elements referenced by guarded iterators.

Parameters

__bi1 First iterator.

__bi2 Second iterator.

Returns

`true` if less.

Definition at line 113 of file `multiway_merge.h`.

5.126.4.2 `template<typename _RAIter, typename _Compare> bool
operator<= (_GuardedIterator< _RAIter, _Compare> & __bi1,
_GuardedIterator< _RAIter, _Compare> & __bi2) [friend]`

Compare two elements referenced by guarded iterators.

Parameters

`__bi1` First iterator.

`__bi2` Second iterator.

Returns

`True` if less equal.

Definition at line 128 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

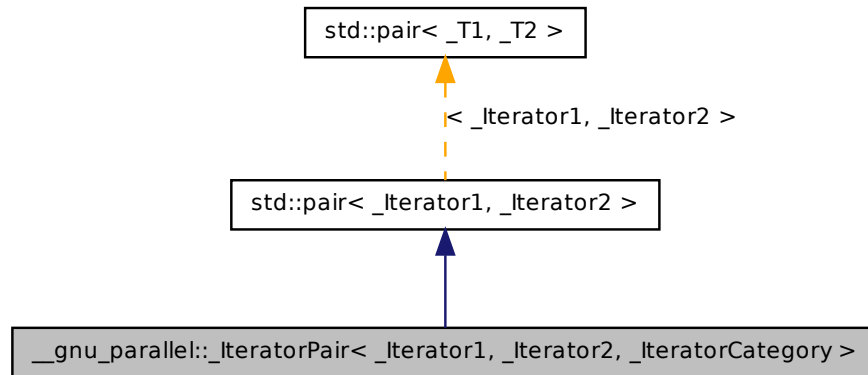
- [multiway_merge.h](#)

5.127 `__gnu_parallel::_IteratorPair< _Iterator1, _ Iterator2, _IteratorCategory> Class Template Reference`

A pair of iterators. The usual iterator operations are applied to both child iterators.

Inheritance diagram for `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _`

IteratorCategory >:



Public Types

- typedef `std::iterator_traits<_Iterator1>` `_TraitsType`
- typedef `_TraitsType::difference_type` `difference_type`
- typedef `_Iterator1` `first_type`
- typedef `_IteratorCategory` `iterator_category`
- typedef `_IteratorPair` * `pointer`
- typedef `_IteratorPair` & `reference`
- typedef `_Iterator2` `second_type`
- typedef void `value_type`

Public Member Functions

- `_IteratorPair` (`const _Iterator1 &__first, const _Iterator2 &__second`)
- `operator _Iterator2` () const
- `_IteratorPair operator+` (`difference_type __delta`) const
- `const _IteratorPair operator++` (int)
- `_IteratorPair & operator++` ()
- `difference_type operator-` (`const _IteratorPair &__other`) const
- `const _IteratorPair operator--` (int)
- `_IteratorPair & operator--` ()

- [_IteratorPair](#) & **operator=** (const [_IteratorPair](#) &__other)
- void **swap** (pair &__p)

Public Attributes

- [_Iterator1](#) [first](#)
- [_Iterator2](#) [second](#)

5.127.1 Detailed Description

```
template<typename _Iterator1,    typename _Iterator2,    typename _-
IteratorCategory> class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2,
_IteratorCategory >
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file iterator.h.

5.127.2 Member Typedef Documentation

5.127.2.1 `typedef _Iterator1 std::pair< _Iterator1 , _Iterator2 >::first_type`
[[inherited](#)]

`first_type` is the first bound type

Definition at line 83 of file stl_pair.h.

5.127.2.2 `typedef _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second_type`
[[inherited](#)]

`second_type` is the second bound type

Definition at line 84 of file stl_pair.h.

5.127.3 Member Data Documentation

5.127.3.1 `_Iterator1 std::pair< _Iterator1 , _Iterator2 >::first` [[inherited](#)]

`first` is a copy of the first object

Definition at line 86 of file stl_pair.h.

**5.127.3.2 `_Iterator2 std::pair<_Iterator1, _Iterator2>::second`
[`inherited`]**

`second` is a copy of the second object

Definition at line 87 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

5.128 `__gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Public Types

- typedef `std::iterator_traits<_Iterator1>::difference_type` **difference_type**
- typedef `_IteratorCategory` **iterator_category**
- typedef `_IteratorTriple` * **pointer**
- typedef `_IteratorTriple` & **reference**
- typedef void **value_type**

Public Member Functions

- `_IteratorTriple` (`const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third`)
- `operator _Iterator3` () `const`
- `_IteratorTriple operator+` (`difference_type __delta`) `const`
- `const _IteratorTriple operator++` (`int`)
- `_IteratorTriple & operator++` ()
- `difference_type operator-` (`const _IteratorTriple &__other`) `const`
- `_IteratorTriple & operator--` ()
- `const _IteratorTriple operator--` (`int`)
- `_IteratorTriple & operator=` (`const _IteratorTriple &__other`)

Public Attributes

- `_Iterator1 _M_first`
- `_Iterator2 _M_second`
- `_Iterator3 _M_third`

5.128.1 Detailed Description

`template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory> class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >`

A triple of iterators. The usual iterator operations are applied to all three child iterators.
Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

5.129 `__gnu_parallel::_Job< _DifferenceTp > Struct` Template Reference

One `__job` for a certain thread.

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Attributes

- `volatile _DifferenceType _M_first`
- `volatile _DifferenceType _M_last`
- `volatile _DifferenceType _M_load`

5.129.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Job< _DifferenceTp >`

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

5.129.2 Member Data Documentation

5.129.2.1 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

5.129.2.2 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_last`

Last element.

Changed by owning thread only.

Definition at line 67 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

5.129.2.3 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_load`

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

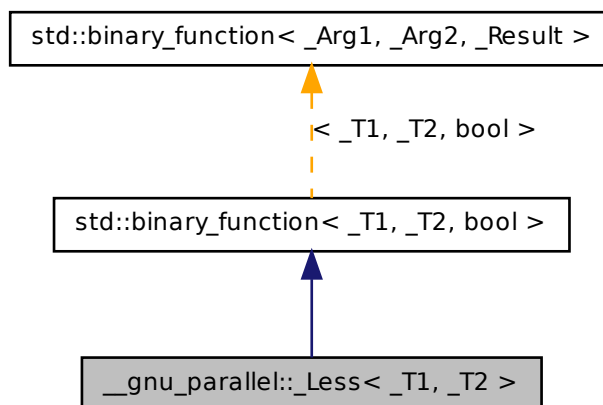
The documentation for this struct was generated from the following file:

- [workstealing.h](#)

5.130 `__gnu_parallel::_Less<_T1, _T2>` Struct Template Reference

Similar to `std::less`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

5.130.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_Less<_T1, _T2>
```

Similar to [std::less](#), but allows two different types.

Definition at line 252 of file `parallel/base.h`.

5.130.2 Member Typedef Documentation

5.130.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.130.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.130.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

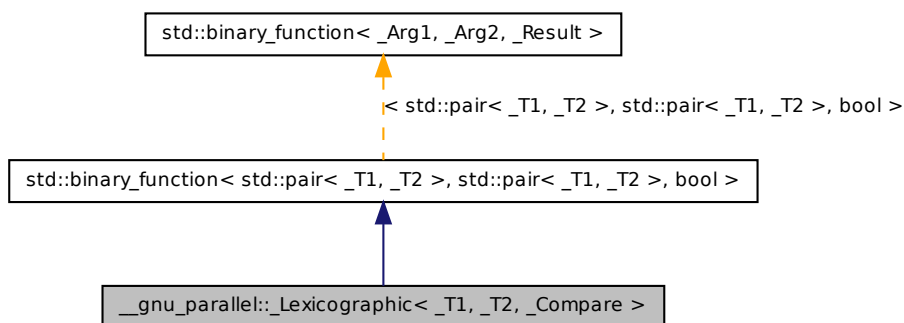
The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.131 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference

Compare __a pair of types lexicographically, ascending.

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`:



Public Types

- typedef `std::pair<_T1, _T2>` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2>` `second_argument_type`

Public Member Functions

- `_Lexicographic` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

5.131.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`

Compare __a pair of types lexicographically, ascending.

Definition at line 55 of file `multiseq_selection.h`.

5.131.2 Member Typedef Documentation

5.131.2.1 `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::first_argument_type`
[inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.131.2.2 `typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::result_type` [inherited]

type of the return type

Definition at line 118 of file stl_function.h.

5.131.2.3 `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::second_argument_type`
[inherited]

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this class was generated from the following file:

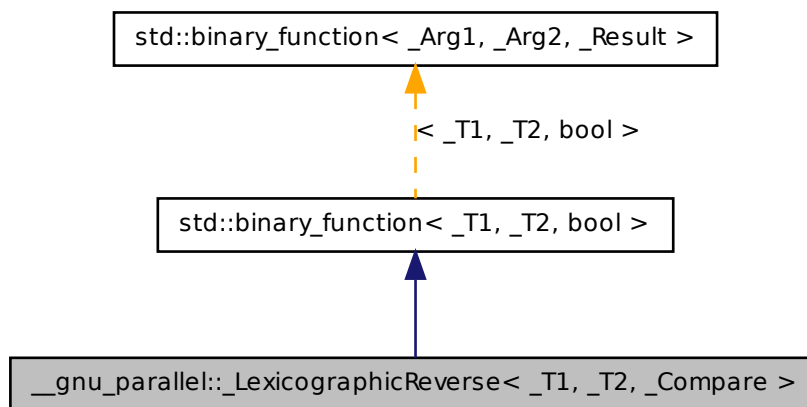
- [multiseq_selection.h](#)

5.132 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference

Compare __a pair of types lexicographically, descending.

Inheritance diagram for __gnu_parallel::_LexicographicReverse< _T1, _T2, _-

Compare >:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `_LexicographicReverse` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

5.132.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>
```

Compare __a pair of types lexicographically, descending.

Definition at line 82 of file `multiseq_selection.h`.

5.132.2 Member Typedef Documentation

5.132.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.132.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.132.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

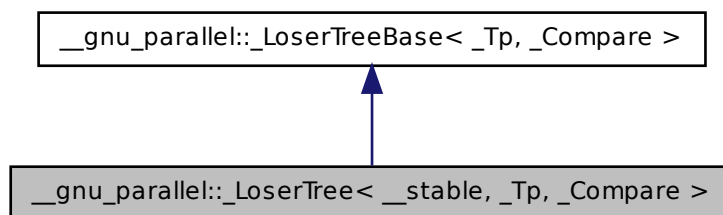
The documentation for this class was generated from the following file:

- [multiseq_selection.h](#)

5.133 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference

Stable [_LoserTree](#) variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTree` (unsigned int __k, _Compare __comp)
- void `__delete_min_insert` (_Tp __key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

5.133.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare> class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`

Stable `_LoserTree` variant. Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

5.133 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference 1161

Unstable variant is done using partial specialisation below.

Definition at line 165 of file `losertree.h`.

5.133.2 Member Function Documentation

5.133.2.1 `template<bool __stable, typename _Tp, typename _Compare
> void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare
>::__delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 217 of file `losertree.h`.

References `std::swap()`.

5.133.2.2 `template<typename _Tp, typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::__get_min_source () [inline, inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 151 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

5.133.2.3 `template<typename _Tp, typename _Compare > void
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start
(const _Tp & __key, int __source, bool __sup) [inline,
inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

`__key` the element to insert

`__source` index of the `__source` sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

5.133.3 Member Data Documentation

5.133.3.1 `template<typename _Tp, typename _Compare > _Compare
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp
[protected, inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

5.133.3.2 `template<typename _Tp, typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_M_first_insert [protected, inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.133.3.3 `template<typename _Tp, typename _Compare > unsigned int
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k
[protected, inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.133.3.4 `template<typename _Tp, typename _Compare > _Loser*
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers
[protected, inherited]`

`_LoserTree` `__elements`.

5.134 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference 1163

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

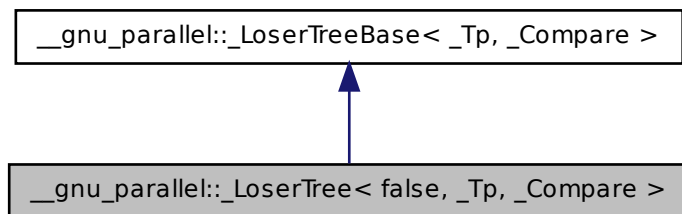
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.134 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Unstable [_LoserTree](#) variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

Protected Attributes

- [_Compare](#) [_M_comp](#)
- [bool](#) [_M_first_insert](#)
- [unsigned int](#) [_M_ik](#)
- [unsigned int](#) [_M_k](#)
- [unsigned int](#) [_M_log_k](#)
- [_Loser](#) * [_M_losers](#)
- [unsigned int](#) [_M_offset](#)

5.134.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable [_LoserTree](#) variant. Stability (non-stable here) is selected with partial specialization.

Definition at line 255 of file losertree.h.

5.134.2 Member Function Documentation

5.134.2.1 `template<typename _Tp , typename _Compare > void
__gnu_parallel::_LoserTree< false, _Tp, _Compare
>::__delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

Parameters

`__key` the `_M_key` to insert

`__sup` true iff `__key` is an explicitly marked supremum

Definition at line 317 of file losertree.h.

References `std::swap()`.

5.134.2.2 `template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::__get_min_source () [inline, inherited]`

Returns

the index of the sequence with the smallest element.

5.134 __gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference 1165

Definition at line 151 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source.

5.134.2.3 `template<typename _Tp , typename _Compare > unsigned int
__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner
(unsigned int __root) [inline]`

Computes the winner of the competition at position "__root".

Called recursively (starting at 0) to build the initial tree.

Parameters

`__root` __index of the "game" to start.

Definition at line 277 of file losertree.h.

5.134.2.4 `template<typename _Tp , typename _Compare > void
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start
(const _Tp & __key, int __source, bool __sup) [inline,
inherited]`

Initializes the sequence "_M_source" with the element "__key".

Parameters

`__key` the element to insert

`__source` __index of the __source __sequence

`__sup` flag that determines whether the value to insert is an explicit __supremum.

Definition at line 130 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup.

5.134.3 Member Data Documentation

5.134.3.1 `template<typename _Tp , typename _Compare > _Compare
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp
[protected, inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

5.134.3.2 `template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_M_first_insert [protected, inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`,
and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.134.3.3 `template<typename _Tp , typename _Compare > unsigned int
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k
[protected, inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.134.3.4 `template<typename _Tp , typename _Compare > _Loser*
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers
[protected, inherited]`

`_LoserTree` __elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

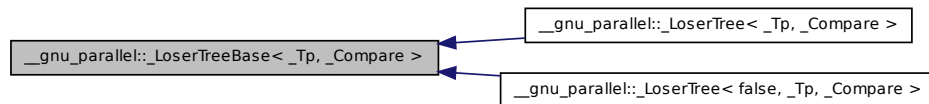
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.135 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Guarded loser/tournament tree.

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



Classes

- struct [_Loser](#)
Internal representation of a [_LoserTree](#) element.

Public Member Functions

- [_LoserTreeBase](#) (unsigned int __k, _Compare __comp)
- [~_LoserTreeBase](#) ()
- int [__get_min_source](#) ()
- void [__insert_start](#) (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare [_M_comp](#)
- bool [_M_first_insert](#)
- unsigned int [_M_ik](#)
- unsigned int [_M_k](#)
- unsigned int [_M_log_k](#)
- [_Loser](#) * [_M_losers](#)
- unsigned int [_M_offset](#)

5.135.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
```

Guarded loser/tournament tree. The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

Parameters

`_Tp` the element type

`_Compare` the comparator to use, defaults to `std::less<_Tp>`

Definition at line 55 of file `losertree.h`.

5.135.2 Constructor & Destructor Documentation

```
5.135.2.1 template<typename _Tp , typename _Compare >
           __gnu_parallel::_LoserTreeBase< _Tp, _Compare
           >::_LoserTreeBase ( unsigned int __k, _Compare __comp )
           [inline]
```

The constructor.

Parameters

`__k` The number of sequences to merge.

`__comp` The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::_rd_log2()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

```
5.135.2.2 template<typename _Tp , typename _Compare >
           __gnu_parallel::_LoserTreeBase< _Tp, _Compare
           >::~~_LoserTreeBase ( ) [inline]
```

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

5.135.3 Member Function Documentation

5.135.3.1 `template<typename _Tp, typename _Compare >`
`int __gnu_parallel::_LoserTreeBase< _Tp, _Compare`
`>::__get_min_source() [inline]`

Returns

the index of the sequence with the smallest element.

Definition at line 151 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

5.135.3.2 `template<typename _Tp, typename _Compare > void`
`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start`
`(const _Tp & __key, int __source, bool __sup) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

`__key` the element to insert

`__source` index of the `__source` sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

5.135.4 Member Data Documentation

5.135.4.1 `template<typename _Tp, typename _Compare > _Compare`
`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp`
`[protected]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

5.135.4.2 `template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_M_first_insert [protected]`

State flag that determines whether the [_LoserTree](#) is empty.

Only used for building the [_LoserTree](#).

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.135.4.3 `template<typename _Tp , typename _Compare > unsigned int
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k
[protected]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.135.4.4 `template<typename _Tp , typename _Compare > _Loser*
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers
[protected]`

[_LoserTree](#) __elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.136 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference

Internal representation of a [_LoserTree](#) element.

Public Attributes

- [_Tp _M_key](#)
- [int _M_source](#)
- [bool _M_sup](#)

5.136.1 Detailed Description

template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser

Internal representation of a [_LoserTree](#) element.

Definition at line 59 of file losertree.h.

5.136.2 Member Data Documentation

**5.136.2.1 template<typename _Tp , typename _Compare >
 _Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare
 >::_Loser::_M_key**

[_M_key](#) of the element in the [_LoserTree](#).

Definition at line 66 of file losertree.h.

Referenced by [__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start\(\)](#).

**5.136.2.2 template<typename _Tp , typename _Compare >
 int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
 >::_Loser::_M_source**

[__index](#) of the [__source](#) [__sequence](#).

Definition at line 64 of file losertree.h.

Referenced by [__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source\(\)](#), and [__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start\(\)](#).

**5.136.2.3 template<typename _Tp , typename _Compare >
 bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
 >::_Loser::_M_sup**

flag, true iff this is a "maximum" [__sentinel](#).

Definition at line 62 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

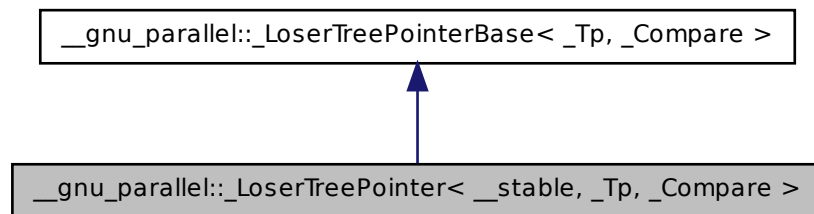
The documentation for this struct was generated from the following file:

- [losertree.h](#)

5.137 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Stable [LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointer` (unsigned int __k, _Compare __comp=[std::less](#)< _Tp >())
- void `__delete_min_insert` (const _Tp &__key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`

5.138 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference 1173

- unsigned int `_M_k`
- [_Loser](#) * `_M_losers`
- unsigned int `_M_offset`

5.137.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`

Stable [_LoserTree](#) implementation. The unstable variant is implemented using partial instantiation below.

Definition at line 401 of file `losertree.h`.

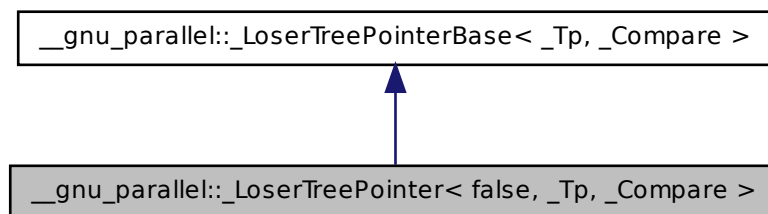
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.138 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Unstable [_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointer` (unsigned int `__k`, `_Compare` `__comp=std::less< _Tp >()`)

- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- [_Loser](#) * **_M_losers**
- unsigned int **_M_offset**

5.138.1 Detailed Description

template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >

Unstable [_LoserTree](#) implementation. The stable variant is above.

Definition at line 482 of file losertree.h.

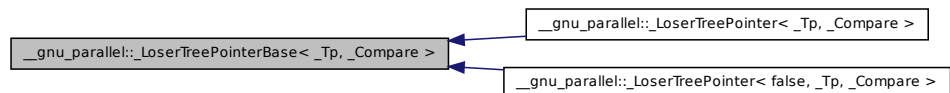
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.139 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference

Base class of [_Loser](#) Tree implementation using pointers.

Inheritance diagram for __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >:



Classes

- struct [_Loser](#)
Internal representation of [_LoserTree](#) __elements.

Public Member Functions

- `_LoserTreePointerBase` (unsigned int __k, _Compare __comp=[std::less](#)<_Tp>())
- int `__get_min_source` ()
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [_Loser](#) * `_M_losers`
- unsigned int `_M_offset`

5.139.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>`

Base class of [_Loser](#) Tree implementation using pointers.

Definition at line 349 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.140 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>::_Loser` Struct Reference

Internal representation of [_LoserTree](#) __elements.

Public Attributes

- `const _Tp * _M_keyp`
- `int _M_source`
- `bool _M_sup`

5.140.1 Detailed Description

`template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`

Internal representation of [_LoserTree](#) __elements.

Definition at line 353 of file `losertree.h`.

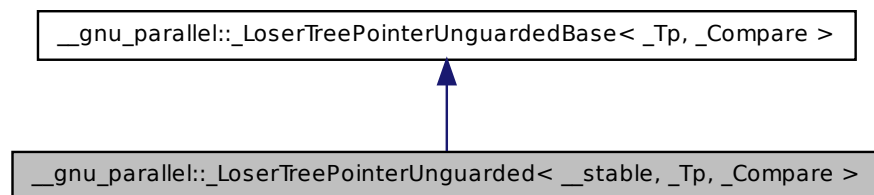
The documentation for this struct was generated from the following file:

- [losertree.h](#)

5.141 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Stable unguarded [_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointerUnguarded` (unsigned int __k, const _Tp &__sentinel, _Compare __comp=`std::less< _Tp >()`)
- void `__delete_min_insert` (const _Tp &__key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

5.141.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`

Stable unguarded [_LoserTree](#) variant storing pointers. Unstable variant is implemented below using partial specialization.

Definition at line 868 of file `losertree.h`.

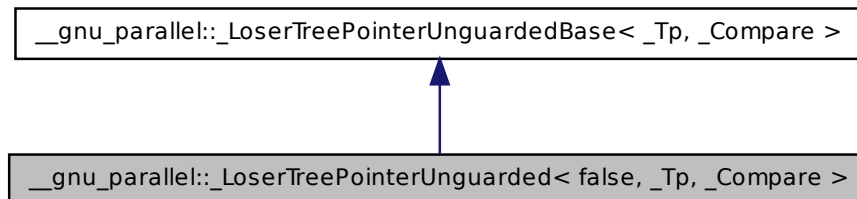
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.142 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Unstable unguarded [_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=`std::less< _Tp >()`)
- void **`_delete_min_insert`** (const _Tp &__key, bool __sup)
- int **`_get_min_source`** ()
- void **`_init`** ()
- unsigned int **`_init_winner`** (unsigned int __root)
- void **`_insert_start`** (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

5.142.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`

Unstable unguarded [_LoserTree](#) variant storing pointers. Stable variant is above.

5.143 `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>` Class Template Reference 1179

Definition at line 953 of file losertree.h.

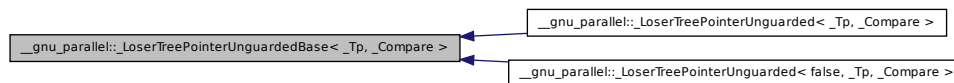
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.143 `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>` Class Template Reference

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>`:



Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int __k, const _Tp &__sentinel, _Compare __comp=`std::less<_Tp>`())
- `int __get_min_source` ()
- `void __insert_start` (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

5.143.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_-
LoserTreePointerUnguardedBase< _Tp, _Compare >
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure. No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 805 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.144 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference

Traits for determining whether the loser tree should use pointers or copies.

Static Public Attributes

- static const bool [_M_use_pointer](#)

5.144.1 Detailed Description

```
template<typename _Tp> struct __gnu_parallel::_LoserTreeTraits< _Tp >
```

Traits for determining whether the loser tree should use pointers or copies. The field "[_M_use_pointer](#)" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false;
};
```

```
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M-
use_pointer = true; };
```


5.145 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference 1181

Parameters

`_Tp` type to give the loser tree traits for.

Definition at line 727 of file `multiway_merge.h`.

5.144.2 Member Data Documentation

5.144.2.1 `template<typename _Tp > const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer`
`[static]`

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

Definition at line 735 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

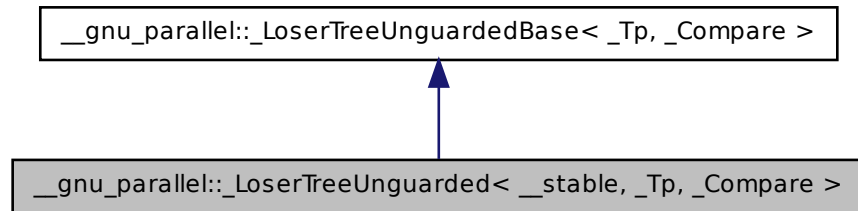
- [multiway_merge.h](#)

5.145 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Stable implementation of unguarded [_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _`

Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp __sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

5.145.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::LoserTreeUnguarded< __stable, _Tp, _Compare >`

Stable implementation of unguarded [_LoserTree](#). Unstable variant is selected below with partial specialization.

5.146 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference 1183

Definition at line 627 of file losertree.h.

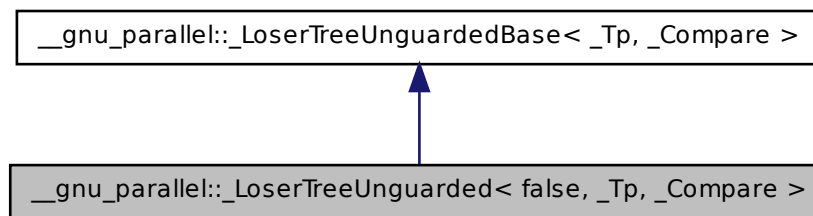
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.146 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Non-Stable implementation of unguarded [_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreeUnguarded` (unsigned int __k, const _Tp __sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void `__delete_min_insert` (_Tp __key, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare _M_comp`

- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

5.146.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`

Non-Stable implementation of unguarded [_LoserTree](#). Stable implementation is above.

Definition at line 713 of file `losertree.h`.

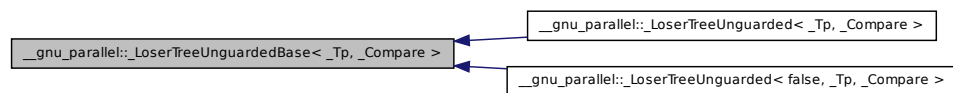
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.147 `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >` Class Template Reference

Base class for unguarded [_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`:



Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int `__k`, const `_Tp` `__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

5.147.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`

Base class for unguarded [_LoserTree](#) implementation. The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

Definition at line 564 of file `losertree.h`.

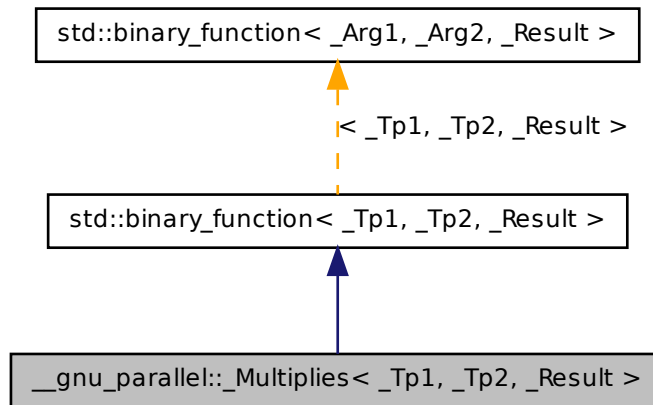
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.148 `__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >` Struct Template Reference

Similar to `std::multiplies`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result >`:



Public Types

- typedef `_Tp1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Tp2` [second_argument_type](#)

Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

5.148.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ -
(*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))> struct __gnu_parallel::_
Multiplies<_Tp1, _Tp2, _Result >
```

Similar to [std::multiplies](#), but allows two different types.

Definition at line 288 of file `parallel/base.h`.

5.148.2 Member Typedef Documentation

5.148.2.1 `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.148.2.2 `typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result
>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.148.2.3 `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.149 `__gnu_parallel::_Nothing` Struct Reference

Functor doing nothing.

Public Member Functions

- `template<typename _It >
void operator\(\) (_It)`

5.149.1 Detailed Description

Functor doing nothing. For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

5.149.2 Member Function Documentation

5.149.2.1 `template<typename _It > void __gnu_parallel::_Nothing::operator()
(_It) [inline]`

Functor execution.

Parameters

`__i` iterator referencing object.

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.150 `__gnu_parallel::_Piece< _DifferenceTp > Struct` Template Reference

Subsequence description.

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Attributes

- `_DifferenceType _M_begin`
- `_DifferenceType _M_end`

5.150.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Piece< _-`
`DifferenceTp >`

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

5.151 `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >` Struct Template Reference

5.150.2 Member Data Documentation

5.150.2.1 `template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

5.150.2.2 `template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end`

End of subsequence.

Definition at line 54 of file `multiway_mergesort.h`.

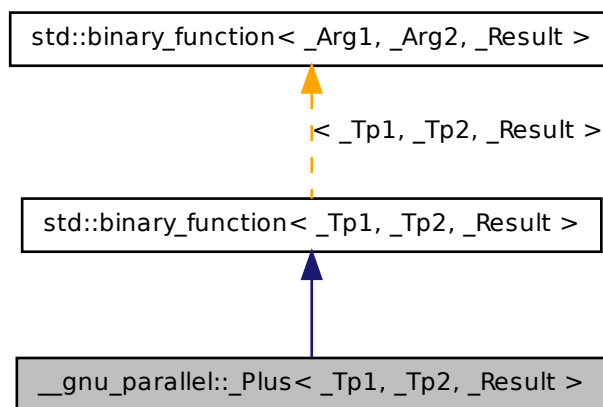
The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.151 `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >` Struct Template Reference

Similar to [std::plus](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >`:



Public Types

- typedef `_Tp1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Tp2` [second_argument_type](#)

Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

5.151.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ -
(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))> struct __gnu_parallel::_
Plus<_Tp1, _Tp2, _Result >
```

Similar to [std::plus](#), but allows two different types.

Definition at line 272 of file `parallel/base.h`.

5.151.2 Member Typedef Documentation

5.151.2.1 `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.151.2.2 `typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result
>::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.151.2.3 `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.152 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference

Data accessed by all threads.

Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator_traits< _RAIter > _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

Public Attributes

- [_ThreadIndex](#) [_M_num_threads](#)
- [_DifferenceType](#) * [_M_offsets](#)

- `std::vector<_Piece<_DifferenceType>>> *_M_pieces`
- `_ValueType *_M_samples`
- `_RAIter _M_source`
- `_DifferenceType *_M_starts`
- `_ValueType **_M_temporary`

5.152.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::PMWMSortingData<_RAIter>`

Data accessed by all threads. PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

5.152.2 Member Data Documentation

5.152.2.1 `template<typename _RAIter> _ThreadIndex
__gnu_parallel::PMWMSortingData<_RAIter>::
_M_num_threads`

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

5.152.2.2 `template<typename _RAIter> _DifferenceType*
__gnu_parallel::PMWMSortingData<_RAIter>::
_M_offsets`

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

5.152.2.3 `template<typename _RAIter> std::vector<_Piece<_
_DifferenceType>>> *_M_pieces`

Pieces of data to merge [thread][__sequence].

Definition at line 86 of file `multiway_mergesort.h`.

5.152 __gnu_parallel::_PMWMSSortingData<_RAIter> Struct Template Reference 1193

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

5.152.2.4 `template<typename _RAIter> _ValueType* __gnu_parallel::_PMWMSSortingData<_RAIter>::_M_samples`

Samples.

Definition at line 80 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), and __gnu_parallel::parallel_sort_mwms().

5.152.2.5 `template<typename _RAIter> _RAIter __gnu_parallel::_PMWMSSortingData<_RAIter>::_M_source`

Input __begin.

Definition at line 71 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

5.152.2.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::_PMWMSSortingData<_RAIter>::_M_starts`

Start indices, per thread.

Definition at line 74 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

5.152.2.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_PMWMSSortingData<_RAIter>::_M_temporary`

Storage in which to sort.

Definition at line 77 of file multiway_mergesort.h.

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.153 `__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >` Class Template Reference

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Public Types

- `typedef _DifferenceTp _DifferenceType`
- `typedef _PseudoSequenceIterator< _Tp, uint64_t > iterator`

Public Member Functions

- `_PseudoSequence (const _Tp &__val, _DifferenceType __count)`
- `iterator begin () const`
- `iterator end () const`

5.153.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

- `_Tp` Sequence `_M` value type.
- `_DifferenceTp` Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

5.153.2 Constructor & Destructor Documentation

```
5.153.2.1 template<typename _Tp, typename _DifferenceTp>
        __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp
        >::_PseudoSequence ( const _Tp & __val, _DifferenceType __count
        ) [inline]
```

Constructor.

5.154 `__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >` Class Template Reference 1195

Parameters

- `_M_val` Element of the sequence.
- `__count` Number of (virtual) copies.

Definition at line 371 of file `parallel/base.h`.

5.153.3 Member Function Documentation

5.153.3.1 `template<typename _Tp, typename _DifferenceTp> iterator
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin () const [inline]`

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

5.153.3.2 `template<typename _Tp, typename _DifferenceTp> iterator
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end () const [inline]`

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.154 `__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >` Class Template Reference

`_Iterator` associated with [__gnu_parallel::_PseudoSequence](#). If features the usual random-access iterator functionality.

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Member Functions

- `_PseudoSequenceIterator` (const `_Tp` &`__val`, `_DifferenceType` `__pos`)

- `bool operator!= (const _PseudoSequenceIterator &__i2)`
- `const _Tp & operator* () const`
- `_PseudoSequenceIterator & operator++ ()`
- `_PseudoSequenceIterator operator++ (int)`
- `_DifferenceType operator- (const _PseudoSequenceIterator &__i2)`
- `bool operator== (const _PseudoSequenceIterator &__i2)`
- `const _Tp & operator[] (_DifferenceType) const`

5.154.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. If features the usual random-access iterator functionality.

Parameters

- `_Tp` Sequence `_M_value` type.
- `DifferenceTp` Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.155 `__gnu_parallel::_QSBThreadLocal`< `_RAIter` > Struct Template Reference

Information local to one thread in the parallel quicksort run.

Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::pair< _RAIter, _RAIter > _Piece`
- `typedef std::iterator_traits< _RAIter > _TraitsType`

Public Member Functions

- `_QSBThreadLocal (int __queue_size)`

Public Attributes

- volatile _DifferenceType * [_M_elements_leftover](#)
- [_Piece _M_global](#)
- [_Piece _M_initial](#)
- [_RestrictedBoundedConcurrentQueue<_Piece> _M_leftover_parts](#)
- [_ThreadIndex _M_num_threads](#)

5.155.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_QSBThreadLocal<_RAIter>`

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

5.155.2 Member Typedef Documentation

5.155.2.1 `template<typename _RAIter> typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_Piece`

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

5.155.3 Constructor & Destructor Documentation

5.155.3.1 `template<typename _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_QSBThreadLocal (int __queue_size) [inline]`

Constructor.

Parameters

`__queue_size` size of the work-stealing queue.

Definition at line 88 of file `balanced_quicksort.h`.

5.155.4 Member Data Documentation

5.155.4.1 `template<typename _RAIter> volatile _DifferenceType*
__gnu_parallel::__QSBThreadLocal< _RAIter
>::__M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.155.4.2 `template<typename _RAIter> _Piece __gnu_parallel::__-
QSBThreadLocal< _RAIter >::__M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

5.155.4.3 `template<typename _RAIter> _Piece __gnu_parallel::__-
QSBThreadLocal< _RAIter >::__M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.155.4.4 `template<typename _RAIter> _-
RestrictedBoundedConcurrentQueue<_Piece>
__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.155.4.5 `template<typename _RAIter> _ThreadIndex
__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced_quicksort.h](#)

5.156 `__gnu_parallel::_RandomNumber` Class Reference

Random number generator, based on the Mersenne twister.

Public Member Functions

- [_RandomNumber](#) ()
- [_RandomNumber](#) (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long [__genrand_bits](#) (int __bits)
- uint32_t [operator\(\)](#) ()
- uint32_t [operator\(\)](#) (uint64_t local_supremum)

5.156.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

5.156.2 Constructor & Destructor Documentation

5.156.2.1 `__gnu_parallel::_RandomNumber::_RandomNumber ()` [[inline](#)]

Default constructor. Seed with 0.

Definition at line 74 of file `random_number.h`.

5.156.2.2 `__gnu_parallel::_RandomNumber::_RandomNumber (uint32_t __seed, uint64_t _M_supremum = 0x100000000ULL)` [[inline](#)]

Constructor.

Parameters

__seed Random __seed.

_M_supremum Generate integer random numbers in the interval [0, *_M_supremum*).

Definition at line 85 of file random_number.h.

5.156.3 Member Function Documentation

5.156.3.1 `unsigned long __gnu_parallel::_RandomNumber::_genrand_bits (int __bits) [inline]`

Generate a number of random bits, run-time parameter.

Parameters

bits Number of bits to generate.

Definition at line 109 of file random_number.h.

5.156.3.2 `uint32_t __gnu_parallel::_RandomNumber::operator() () [inline]`

Generate unsigned random 32-bit integer.

Definition at line 94 of file random_number.h.

5.156.3.3 `uint32_t __gnu_parallel::_RandomNumber::operator() (uint64_t local_supremum) [inline]`

Generate unsigned random 32-bit integer in the interval [0, *local_supremum*).

Definition at line 100 of file random_number.h.

The documentation for this class was generated from the following file:

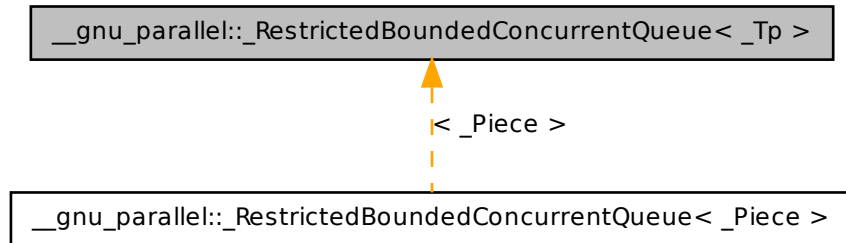
- [random_number.h](#)

5.157 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

Double-ended queue of bounded size, allowing lock-free atomic access. [push_front\(\)](#) and [pop_front\(\)](#) must not be called concurrently to each other, while [pop_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.

5.157 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference 1201

Inheritance diagram for `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`:



Public Member Functions

- `_RestrictedBoundedConcurrentQueue` (`_SequenceIndex` `__max_size`)
- `~_RestrictedBoundedConcurrentQueue` ()
- `bool pop_back` (`_Tp &__t`)
- `bool pop_front` (`_Tp &__t`)
- `void push_front` (`const _Tp &__t`)

5.157.1 Detailed Description

template<typename `_Tp`> **class** `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

`_Tp` Contained element type.

Definition at line 52 of file `queue.h`.

5.157.2 Constructor & Destructor Documentation

5.157.2.1 `template<typename _Tp> __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp
>::_RestrictedBoundedConcurrentQueue (_SequenceIndex
__max_size) [inline]`

Constructor. Not to be called concurrent, of course.

Parameters

__M_max_size Maximal number of elements to be contained.

Definition at line 68 of file queue.h.

5.157.2.2 `template<typename _Tp> __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp
>::~~RestrictedBoundedConcurrentQueue () [inline]`

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

5.157.3 Member Function Documentation

5.157.3.1 `template<typename _Tp> bool __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp >::pop_back (_Tp &
__t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop_front\(\)](#).

Definition at line 127 of file queue.h.

5.157.3.2 `template<typename _Tp> bool __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp >::pop_front (_Tp &
__t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop_front\(\)](#).

Definition at line 100 of file queue.h.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.158 `__gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference 1203

5.157.3.3 `template<typename _Tp> void __gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >::push_front (const _Tp & __t) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with [pop_front\(\)](#).

Definition at line 83 of file [queue.h](#).

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this class was generated from the following file:

- [queue.h](#)

5.158 `__gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Stable sorting functor.

Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

5.158.1 Detailed Description

`template<bool __stable, class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor. Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1004 of file [multiway_merge.h](#).

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.159 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference`

Non-`__stable` sorting functor.

Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

5.159.1 Detailed Description

`template<class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-`__stable` sorting functor. Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1017 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.160 `__gnu_parallel::_Settings Struct Reference`

`class _Settings` /// Run-time settings for the parallel mode including all tunable parameters.

Static Public Member Functions

- `static _GLIBCXX_CONST const _Settings & get () throw ()`
- `static void set (_Settings &) throw ()`

Public Attributes

- `_SequenceIndex accumulate_minimal_n`
- `unsigned int adjacent_difference_minimal_n`
- `_AlgorithmStrategy algorithm_strategy`

- unsigned int [cache_line_size](#)
- [_SequenceIndex](#) count_minimal_n
- [_SequenceIndex](#) fill_minimal_n
- [_FindAlgorithm](#) **find_algorithm**
- double [find_increasing_factor](#)
- [_SequenceIndex](#) find_initial_block_size
- [_SequenceIndex](#) find_maximum_block_size
- float [find_scale_factor](#)
- [_SequenceIndex](#) find_sequential_search_size
- [_SequenceIndex](#) for_each_minimal_n
- [_SequenceIndex](#) generate_minimal_n
- unsigned long long [L1_cache_size](#)
- unsigned long long [L2_cache_size](#)
- [_SequenceIndex](#) max_element_minimal_n
- [_SequenceIndex](#) merge_minimal_n
- unsigned int [merge_oversampling](#)
- [_SplittingAlgorithm](#) **merge_splitting**
- [_SequenceIndex](#) min_element_minimal_n
- [_MultiwayMergeAlgorithm](#) **multiway_merge_algorithm**
- int [multiway_merge_minimal_k](#)
- [_SequenceIndex](#) multiway_merge_minimal_n
- unsigned int [multiway_merge_oversampling](#)
- [_SplittingAlgorithm](#) **multiway_merge_splitting**
- [_SequenceIndex](#) nth_element_minimal_n
- [_SequenceIndex](#) partial_sort_minimal_n
- [_PartialSumAlgorithm](#) **partial_sum_algorithm**
- float [partial_sum_dilation](#)
- unsigned int [partial_sum_minimal_n](#)
- double [partition_chunk_share](#)
- [_SequenceIndex](#) partition_chunk_size
- [_SequenceIndex](#) partition_minimal_n
- [_SequenceIndex](#) qsb_steals
- unsigned int [random_shuffle_minimal_n](#)
- [_SequenceIndex](#) replace_minimal_n
- [_SequenceIndex](#) search_minimal_n
- [_SequenceIndex](#) set_difference_minimal_n
- [_SequenceIndex](#) set_intersection_minimal_n
- [_SequenceIndex](#) set_symmetric_difference_minimal_n
- [_SequenceIndex](#) set_union_minimal_n
- [_SortAlgorithm](#) **sort_algorithm**
- [_SequenceIndex](#) sort_minimal_n
- unsigned int [sort_mwms_oversampling](#)
- unsigned int [sort_qs_num_samples_preset](#)

- [_SequenceIndex](#) `sort_qsb_base_case_maximal_n`
- [_SplittingAlgorithm](#) `sort_splitting`
- unsigned int `TLB_size`
- [_SequenceIndex](#) `transform_minimal_n`
- [_SequenceIndex](#) `unique_copy_minimal_n`
- [_SequenceIndex](#) `workstealing_chunk_size`

5.160.1 Detailed Description

class [_Settings](#) /// Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file settings.h.

5.160.2 Member Function Documentation

5.160.2.1 static [_GLIBCXX_CONST](#) const [_Settings&](#) [__gnu_parallel::_Settings::get](#) () throw () [**static**]

Get the global settings.

Referenced by [__gnu_parallel::__find_template\(\)](#), [__gnu_parallel::__for_each_template_random_access_workstealing\(\)](#), [__gnu_parallel::__parallel_nth_element\(\)](#), [__gnu_parallel::__parallel_partial_sum\(\)](#), [__gnu_parallel::__parallel_partial_sum_linear\(\)](#), [__gnu_parallel::__parallel_partition\(\)](#), [__gnu_parallel::__parallel_sort\(\)](#), [__gnu_parallel::__parallel_sort_qs_conquer\(\)](#), [__gnu_parallel::__qsb_local_sort_with_helping\(\)](#), [__gnu_parallel::multiway_merge_sampling_splitting\(\)](#), [__gnu_parallel::parallel_multiway_merge\(\)](#), [__gnu_parallel::parallel_sort_mwms\(\)](#), and [__gnu_parallel::parallel_sort_mwms_pu\(\)](#).

5.160.2.2 static void [__gnu_parallel::_Settings::set](#) ([_Settings &](#)) throw () [**static**]

Set the global settings.

5.160.3 Member Data Documentation

5.160.3.1 [_SequenceIndex](#) [__gnu_parallel::_Settings::accumulate_minimal_n](#)

Minimal input size for accumulate.

Definition at line 139 of file settings.h.

5.160.3.2 `unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n`

Minimal input size for `adjacent_difference`.

Definition at line 142 of file `settings.h`.

5.160.3.3 `unsigned int __gnu_parallel::_Settings::cache_line_size`

Overestimation of cache line size. Used to avoid false /// sharing, i.e. elements of different threads are at least this /// amount apart.

Definition at line 265 of file `settings.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

5.160.3.4 `_SequenceIndex __gnu_parallel::_Settings::count_minimal_n`

Minimal input size for `count` and `count_if`.

Definition at line 145 of file `settings.h`.

5.160.3.5 `_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n`

Minimal input size for `fill`.

Definition at line 148 of file `settings.h`.

5.160.3.6 `double __gnu_parallel::_Settings::find_increasing_factor`

Block size increase factor for `find`.

Definition at line 151 of file `settings.h`.

5.160.3.7 `_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size`

Initial block size for `find`.

Definition at line 154 of file `settings.h`.

Referenced by `__gnu_parallel::__find_template()`.

5.160.3.8 `_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size`

Maximal block size for `find`.

Definition at line 157 of file settings.h.

5.160.3.9 float __gnu_parallel::_Settings::find_scale_factor

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by __gnu_parallel::_find_template().

5.160.3.10 _SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by __gnu_parallel::_find_template().

5.160.3.11 _SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n

Minimal input size for for_each.

Definition at line 163 of file settings.h.

5.160.3.12 _SequenceIndex __gnu_parallel::_Settings::generate_minimal_n

Minimal input size for generate.

Definition at line 166 of file settings.h.

5.160.3.13 unsigned long long __gnu_parallel::_Settings::L1_cache_size

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

5.160.3.14 unsigned long long __gnu_parallel::_Settings::L2_cache_size

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by __gnu_parallel::_parallel_random_shuffle_drs(), and __gnu_parallel::_sequential_random_shuffle().

5.160.3.15 _SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n

Minimal input size for max_element.

Definition at line 169 of file settings.h.

5.160.3.16 _SequenceIndex __gnu_parallel::_Settings::merge_minimal_n

Minimal input size for merge.

Definition at line 172 of file settings.h.

5.160.3.17 unsigned int __gnu_parallel::_Settings::merge_oversampling

Oversampling factor for merge.

Definition at line 175 of file settings.h.

Referenced by __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiway_merge().

5.160.3.18 _SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n

Minimal input size for min_element.

Definition at line 178 of file settings.h.

5.160.3.19 int __gnu_parallel::_Settings::multiway_merge_minimal_k

Oversampling factor for multiway_merge.

Definition at line 184 of file settings.h.

5.160.3.20 _SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n

Minimal input size for multiway_merge.

Definition at line 181 of file settings.h.

5.160.3.21 unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling

Oversampling factor for multiway_merge.

Definition at line 187 of file settings.h.

5.160.3.22 _SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n

Minimal input size for nth_element.

Definition at line 190 of file settings.h.

Referenced by __gnu_parallel::_parallel_nth_element().

5.160.3.23 _SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n

Minimal input size for partial_sort.

Definition at line 203 of file settings.h.

5.160.3.24 float __gnu_parallel::_Settings::partial_sum_dilation

Ratio for partial_sum. Assume "sum and write result" to be /// this factor slower than just "sum".

Definition at line 207 of file settings.h.

Referenced by __gnu_parallel::_parallel_partial_sum_linear().

5.160.3.25 unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n

Minimal input size for partial_sum.

Definition at line 210 of file settings.h.

5.160.3.26 double __gnu_parallel::_Settings::partition_chunk_share

Chunk size for partition, relative to input size. If > 0.0, /// this value overrides partition_chunk_size.

Definition at line 197 of file settings.h.

Referenced by __gnu_parallel::_parallel_partition().

5.160.3.27 _SequenceIndex __gnu_parallel::_Settings::partition_chunk_size

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by __gnu_parallel::__parallel_partition().

5.160.3.28 _SequenceIndex __gnu_parallel::_Settings::partition_minimal_n

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by __gnu_parallel::__parallel_nth_element().

5.160.3.29 _SequenceIndex __gnu_parallel::_Settings::qsb_steals

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

5.160.3.30 unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n

Minimal input size for random_shuffle.

Definition at line 213 of file settings.h.

5.160.3.31 _SequenceIndex __gnu_parallel::_Settings::replace_minimal_n

Minimal input size for replace and replace_if.

Definition at line 216 of file settings.h.

5.160.3.32 _SequenceIndex __gnu_parallel::_Settings::search_minimal_n

Minimal input size for search and search_n.

Definition at line 273 of file settings.h.

5.160.3.33 _SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n

Minimal input size for set_difference.

Definition at line 219 of file settings.h.

5.160.3.34 `_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

5.160.3.35 `_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

5.160.3.36 `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

5.160.3.37 `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

5.160.3.38 `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

5.160.3.39 `unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

5.161 __gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference 1213

5.160.3.40 _SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n

Maximal subsequence __length to switch to unbalanced __base case. /// Applies to std::sort with dynamically load-balanced quicksort.

Definition at line 241 of file settings.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

5.160.3.41 unsigned int __gnu_parallel::_Settings::TLB_size

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file settings.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__sequential_random_shuffle().

5.160.3.42 _SequenceIndex __gnu_parallel::_Settings::transform_minimal_n

Minimal input size for parallel std::transform.

Definition at line 244 of file settings.h.

5.160.3.43 _SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n

Minimal input size for unique_copy.

Definition at line 247 of file settings.h.

The documentation for this struct was generated from the following file:

- [settings.h](#)

5.161 __gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

Split consistently.

5.161.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< __exact, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.162 __gnu_parallel::_SplitConsistently< false, _-RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

Split by sampling.

Public Member Functions

- void **operator()** (const [_ThreadIndex](#) __iam, [_PMWMSortingData](#)< _RAIter > *__sd, _Compare &__comp, const typename [std::iterator_traits](#)< _RAIter >::difference_type __num_samples) const

5.162.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< false,
_RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.163 `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference 1215

5.163 `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

Split by exact splitting.

Public Member Functions

- void **operator()** (const [_ThreadIndex](#) __iam, [_PMWMSSortingData](#)< _RAIter > *__sd, _Compare &__comp, const typename [std::iterator_traits](#)< _RAIter >::difference_type __num_samples) const

5.163.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::SplitConsistently< true, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

Definition at line 128 of file `multiway_mergesort.h`.

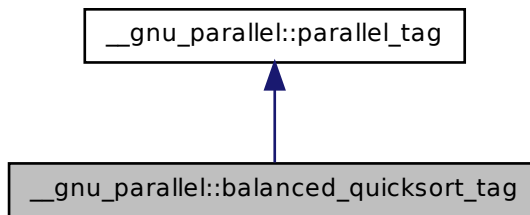
The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.164 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Forces parallel sorting using balanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



Public Member Functions

- `balanced_quicksort_tag` (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- `void` `set_num_threads` (`_ThreadIndex` __num_threads)

5.164.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

5.164.2 Member Function Documentation

5.164.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.164.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

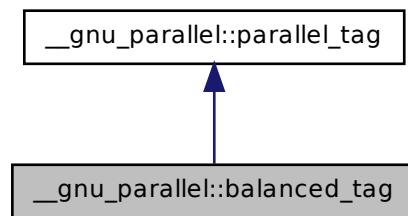
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.165 __gnu_parallel::balanced_tag Struct Reference

Recommends parallel execution using dynamic load-balancing at compile time.

Inheritance diagram for __gnu_parallel::balanced_tag:



Public Member Functions

- [_ThreadIndex](#) [__get_num_threads](#) ()
- void [set_num_threads](#) ([_ThreadIndex](#) __num_threads)

5.165.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

5.165.2 Member Function Documentation

5.165.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.165.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

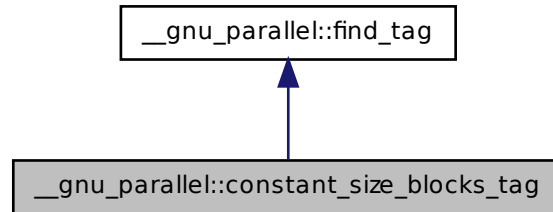
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.166 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Selects the constant block size variant for `std::find()`.

Inheritance diagram for __gnu_parallel::constant_size_blocks_tag:



5.166.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

[_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS](#)

Definition at line 178 of file `tags.h`.

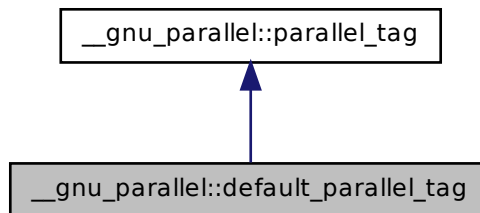
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.167 __gnu_parallel::default_parallel_tag Struct Reference

Recommends parallel execution using the default parallel algorithm.

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



Public Member Functions

- `default_parallel_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.167.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file tags.h.

5.167.2 Member Function Documentation

5.167.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.167.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

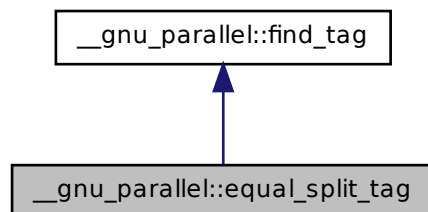
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.168 __gnu_parallel::equal_split_tag Struct Reference

Selects the equal splitting variant for std::find().

Inheritance diagram for __gnu_parallel::equal_split_tag:



5.168.1 Detailed Description

Selects the equal splitting variant for std::find().

See also

[_GLIBCXX_FIND_EQUAL_SPLIT](#)

Definition at line 182 of file tags.h.

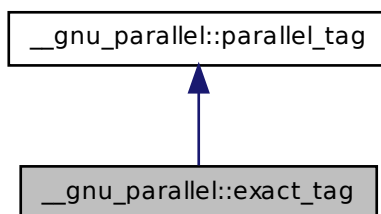
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.169 __gnu_parallel::exact_tag Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for __gnu_parallel::exact_tag:



Public Member Functions

- `exact_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.169.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

5.169.2 Member Function Documentation

5.169.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [`inline`, `inherited`]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.169.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [`inline`, `inherited`]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

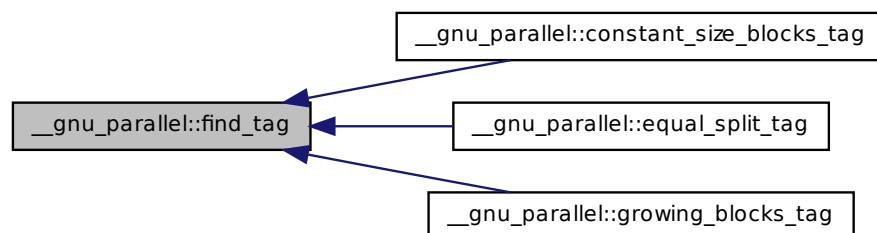
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.170 `__gnu_parallel::find_tag` Struct Reference

Base class for `std::find()` variants.

Inheritance diagram for `__gnu_parallel::find_tag`:



5.170.1 Detailed Description

Base class for `std::find()` variants.

Definition at line 104 of file `tags.h`.

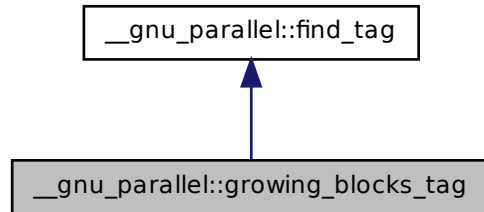
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.171 `__gnu_parallel::growing_blocks_tag` Struct Reference

Selects the growing block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



5.171.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

[_GLIBCXX_FIND_GROWING_BLOCKS](#)

Definition at line 174 of file `tags.h`.

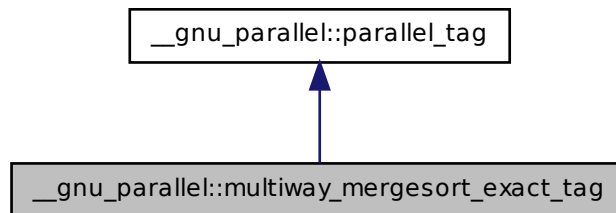
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.172 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



Public Member Functions

- `multiway_mergesort_exact_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.172.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

5.172.2 Member Function Documentation

5.172.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.173 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference 1227

5.172.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

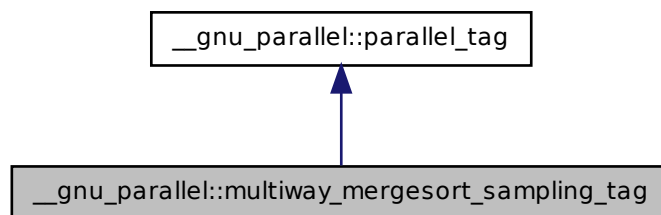
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.173 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



Public Member Functions

- `multiway_mergesort_sampling_tag (_ThreadIndex __num_threads)`
- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

5.173.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

5.173.2 Member Function Documentation

5.173.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.173.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

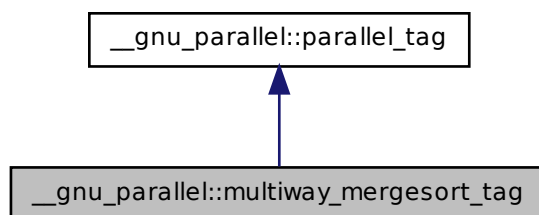
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.174 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Forces parallel sorting using multiway mergesort at compile time.

Inheritance diagram for __gnu_parallel::multiway_mergesort_tag:



Public Member Functions

- `multiway_mergesort_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.174.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

5.174.2 Member Function Documentation

5.174.2.1 [_ThreadIndex](#) __gnu_parallel::parallel_tag::__get_num_threads () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.174.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

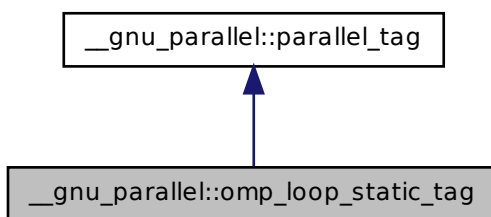
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.175 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Recommends parallel execution using OpenMP static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



Public Member Functions

- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

5.175.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

5.175.2 Member Function Documentation

5.175.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.175.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

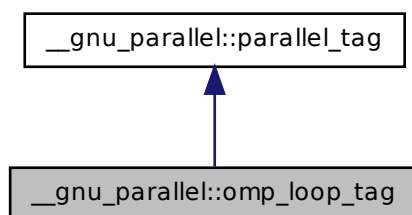
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.176 `__gnu_parallel::omp_loop_tag` Struct Reference

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads\(\)](#)
- void [set_num_threads\(_ThreadIndex __num_threads\)](#)

5.176.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

5.176.2 Member Function Documentation

5.176.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.176.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
 __num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

 __num_threads Desired number of threads.

Definition at line 73 of file tags.h.

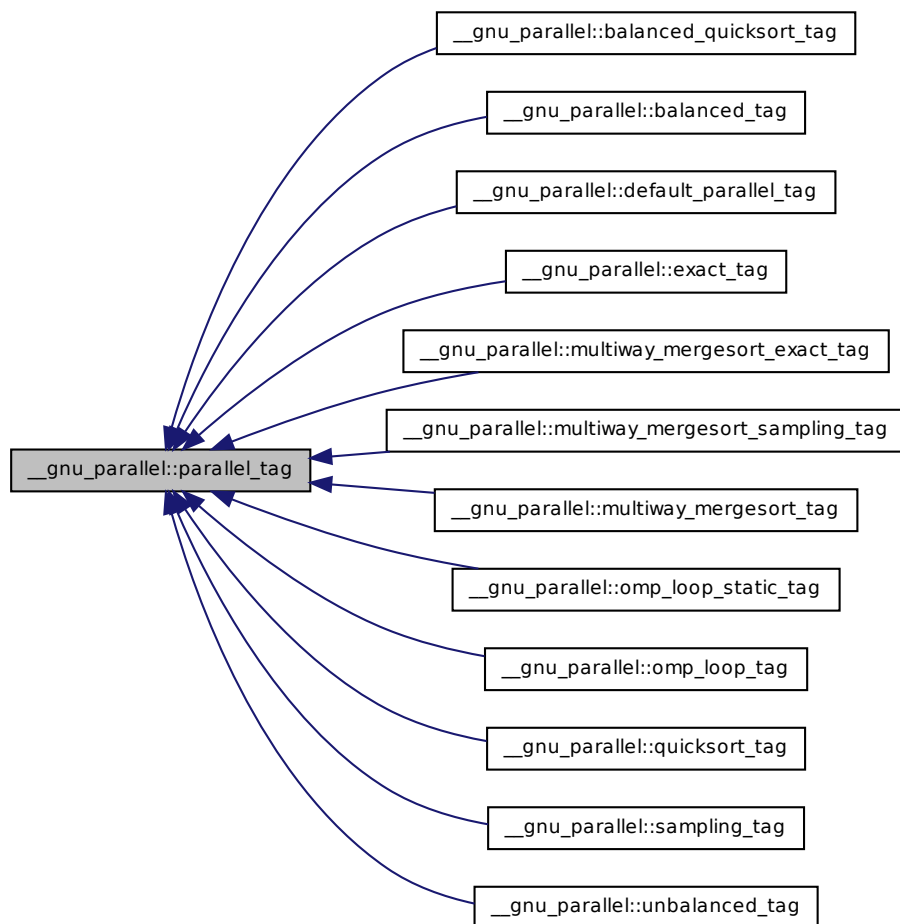
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.177 __gnu_parallel::parallel_tag Struct Reference

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Inheritance diagram for `__gnu_parallel::parallel_tag`:



Public Member Functions

- [parallel_tag](#) ()
- [parallel_tag](#) ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) [__get_num_threads](#) ()
- void [set_num_threads](#) ([_ThreadIndex](#) __num_threads)

5.177.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

5.177.2 Constructor & Destructor Documentation

5.177.2.1 __gnu_parallel::parallel_tag::parallel_tag () [inline]

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

5.177.2.2 __gnu_parallel::parallel_tag::parallel_tag (_ThreadIndex __num_threads) [inline]

Default constructor. Recommend number of threads to use.

Parameters

__num_threads Desired number of threads.

Definition at line 58 of file tags.h.

5.177.3 Member Function Documentation

5.177.3.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

5.177.3.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

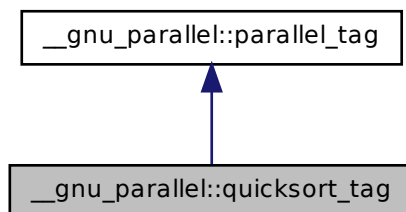
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.178 `__gnu_parallel::quicksort_tag` Struct Reference

Forces parallel sorting using unbalanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::quicksort_tag`:

**Public Member Functions**

- `quicksort_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void [set_num_threads](#) ([_ThreadIndex](#) __num_threads)

5.178.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

5.178.2 Member Function Documentation

5.178.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

5.178.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

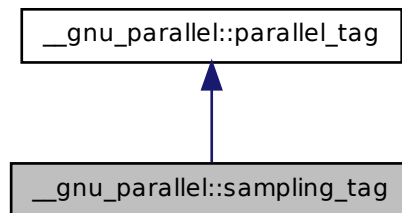
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.179 __gnu_parallel::sampling_tag Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for `__gnu_parallel::sampling_tag`:



Public Member Functions

- `sampling_tag` (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` __num_threads)

5.179.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

5.179.2 Member Function Documentation

5.179.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.179.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

5.180 `__gnu_parallel::sequential_tag` Struct Reference

Forces sequential execution at compile time.

5.180.1 Detailed Description

Forces sequential execution at compile time.

Definition at line 42 of file `tags.h`.

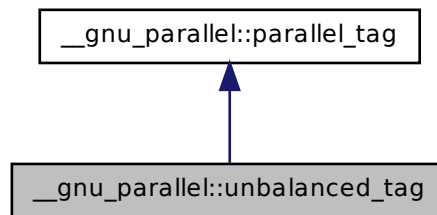
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.181 `__gnu_parallel::unbalanced_tag` Struct Reference

Recommends parallel execution using static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads \(\)](#)
- void [set_num_threads \(_ThreadIndex __num_threads\)](#)

5.181.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file `tags.h`.

5.181.2 Member Function Documentation

5.181.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.181.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

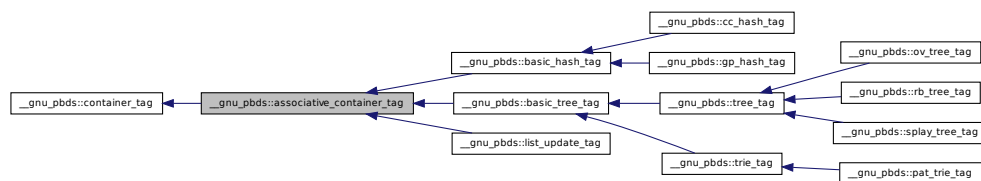
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.182 `__gnu_pbds::associative_container_tag` Struct Reference

Basic associative-container.

Inheritance diagram for `__gnu_pbds::associative_container_tag`:



5.182.1 Detailed Description

Basic associative-container.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.183 **__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator > Class Template Reference**

An abstract basic hash-based associative container.

Inheritance diagram for __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

5.183.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn,
typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_TL,
typename Allocator> class __gnu_pbds::basic_hash_table< Key, Mapped,
Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >
```

An abstract basic hash-based associative container.

Definition at line 144 of file `assoc_container.hpp`.

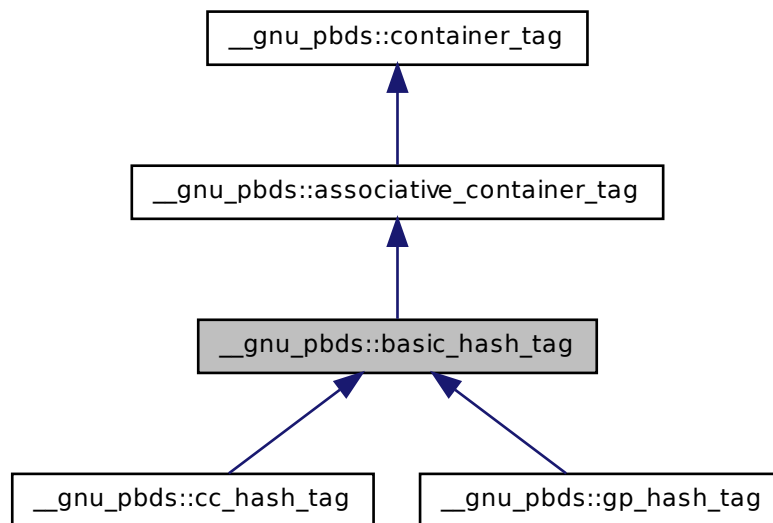
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.184 __gnu_pbds::basic_hash_tag Struct Reference

Basic hash.

Inheritance diagram for __gnu_pbds::basic_hash_tag:



5.184.1 Detailed Description

Basic hash.

Definition at line 106 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.185 `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >` Class Template Reference

An abstract basic tree-like (tree, trie) associative container.

Inheritance diagram for `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**

- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef Node_Update **node_update**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

5.185.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename Allocator> class `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`

An abstract basic tree-like (tree, trie) associative container.

Definition at line 477 of file `assoc_container.hpp`.

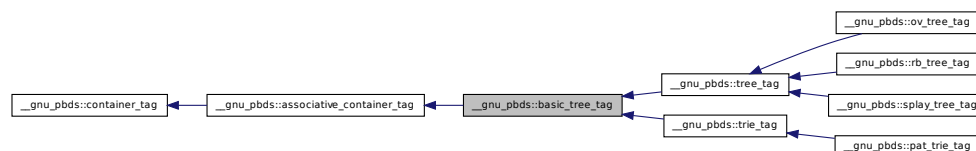
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.186 `__gnu_pbds::basic_tree_tag` Struct Reference

Basic tree.

Inheritance diagram for `__gnu_pbds::basic_tree_tag`:



5.186.1 Detailed Description

Basic tree.

Definition at line 115 of file tag_and_trait.hpp.

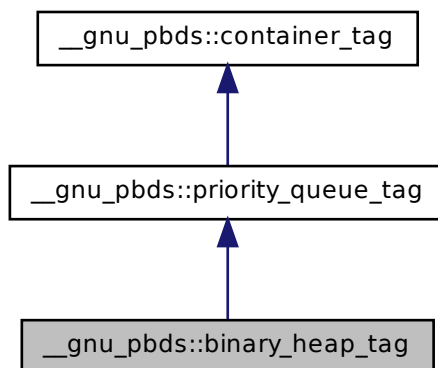
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.187 `__gnu_pbds::binary_heap_tag` Struct Reference

Binary-heap (array-based).

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



5.187.1 Detailed Description

Binary-heap (array-based).

Definition at line 151 of file tag_and_trait.hpp.

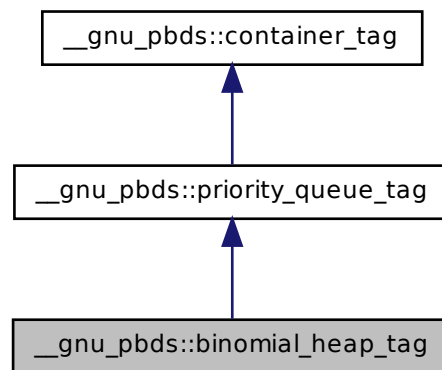
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.188 __gnu_pbds::binomial_heap_tag Struct Reference

Binomial-heap.

Inheritance diagram for __gnu_pbds::binomial_heap_tag:



5.188.1 Detailed Description

Binomial-heap.

Definition at line 145 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.189 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >` Class Template Reference

A concrete collision-chaining hash-based associative container.

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef [cc_hash_tag](#) **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**

5.189 `__gnu_pbds::cc_hash_table`< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template

Reference

1249

- `typedef Resize_Policy resize_policy`
- `typedef allocator_type::size_type size_type`
- `typedef allocator_type::template rebind< value_type >::other value_rebind`
- `typedef base_type::value_type value_type`

Public Member Functions

- `cc_hash_table (const hash_fn &h)`
- `cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)`
- `template<typename It >`
`cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)`
- `cc_hash_table (const cc_hash_table &other)`
- `template<typename It >`
`cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)`
- `template<typename It >`
`cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)`
- `cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)`
- `template<typename It >`
`cc_hash_table (It first, It last, const hash_fn &h)`
- `template<typename It >`
`cc_hash_table (It first, It last)`
- `cc_hash_table (const hash_fn &h, const eq_fn &e)`
- `cc_hash_table & operator= (const cc_hash_table &other)`
- `void swap (cc_hash_table &other)`

5.189.1 Detailed Description

`template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename Allocator = std::allocator<char>>> class __gnu_pbds::cc_hash_table<Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`

A concrete collision-chaining hash-based associative container.

Definition at line 180 of file `assoc_container.hpp`.

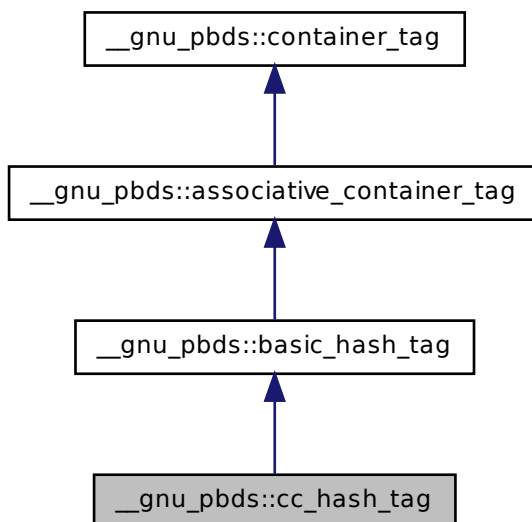
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.190 __gnu_pbds::cc_hash_tag Struct Reference

Collision-chaining hash.

Inheritance diagram for __gnu_pbds::cc_hash_tag:



5.190.1 Detailed Description

Collision-chaining hash.

Definition at line 109 of file `tag_and_trait.hpp`.

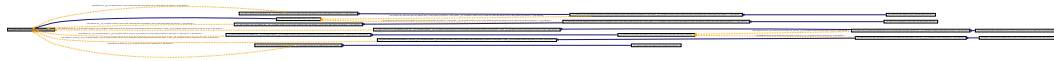
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.191 `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >` Class Template Reference

An abstract basic associative container.

Inheritance diagram for `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

5.191.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Policy_Tl, typename Allocator> class __gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >

An abstract basic associative container.

Definition at line 77 of file `assoc_container.hpp`.

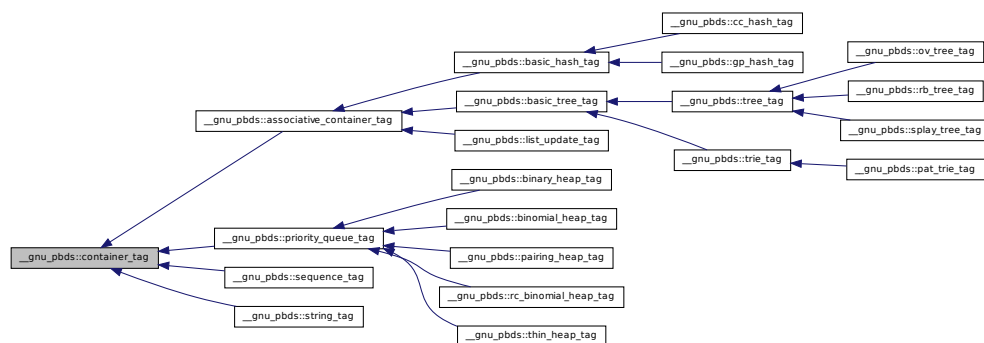
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.192 __gnu_pbds::container_tag Struct Reference

Base data structure tag.

Inheritance diagram for `__gnu_pbds::container_tag`:



5.192.1 Detailed Description

Base data structure tag.

Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.193 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

[container_traits](#)

Inherits `container_traits_base< Cntnr::container_category >`.

Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `container_traits_base< container_category >` **base_type**
- typedef `Cntnr::container_category` **container_category**
- typedef `Cntnr` **container_type**
- typedef `base_type::invalidation_guarantee` **invalidation_guarantee**

5.193.1 Detailed Description

`template<typename Cntnr> struct __gnu_pbds::container_traits< Cntnr >`

[container_traits](#)

Definition at line 346 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.194 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >` Struct Template Reference

Public Types

- typedef `mapped_type_allocator::const_pointer` **const_mapped_pointer**
- typedef `mapped_type_allocator::const_reference` **const_mapped_reference**
- typedef `value_type_allocator::const_pointer` **const_pointer**
- typedef `value_type_allocator::const_reference` **const_reference**
- typedef `mapped_type_allocator::pointer` **mapped_pointer**
- typedef `mapped_type_allocator::reference` **mapped_reference**
- typedef `mapped_type_allocator::value_type` **mapped_type**

- `typedef Allocator::template rebind< Mapped >::other mapped_type_allocator`
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef value_type_allocator::value_type value_type`
- `typedef Allocator::template rebind< std::pair< const Key, Mapped > >::other value_type_allocator`

5.194.1 Detailed Description

`template<typename Key, typename Mapped, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 61 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.195 `__gnu_pbds::detail::value_type_base`< Key, Mapped, Allocator, true > Struct Template Reference

Public Types

- `typedef mapped_type_allocator::const_pointer const_mapped_pointer`
- `typedef mapped_type_allocator::const_reference const_mapped_reference`
- `typedef value_type_allocator::const_pointer const_pointer`
- `typedef value_type_allocator::const_reference const_reference`
- `typedef mapped_type_allocator::pointer mapped_pointer`
- `typedef mapped_type_allocator::reference mapped_reference`
- `typedef mapped_type_allocator::value_type mapped_type`
- `typedef Allocator::template rebind< Mapped >::other mapped_type_allocator`
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef value_type_allocator::value_type value_type`
- `typedef Allocator::template rebind< std::pair< const Key, Mapped > >::other value_type_allocator`

5.195.1 Detailed Description

template<typename Key, typename Mapped, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >

Specialization of value_type_base for the case where the hash value is stored alongside each value.

Definition at line 88 of file basic_types.hpp.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.196 __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false > Struct Template Reference

Public Types

- typedef mapped_type_allocator::const_pointer **const_mapped_pointer**
- typedef mapped_type_allocator::const_reference **const_mapped_reference**
- typedef value_type_allocator::const_pointer **const_pointer**
- typedef value_type_allocator::const_reference **const_reference**
- typedef mapped_type_allocator::pointer **mapped_pointer**
- typedef mapped_type_allocator::reference **mapped_reference**
- typedef mapped_type_allocator::value_type **mapped_type**
- typedef Allocator::template rebind< [null_mapped_type](#) >::other **mapped_type_allocator**
- typedef value_type_allocator::pointer **pointer**
- typedef value_type_allocator::reference **reference**
- typedef Key **value_type**
- typedef Allocator::template rebind< value_type >::other **value_type_allocator**

Static Public Attributes

- static [null_mapped_type](#) **s_null_mapped**

5.196.1 Detailed Description

```
template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >
```

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 122 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.197 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >` Struct Template Reference

Public Types

- `typedef mapped_type_allocator::const_pointer const_mapped_pointer`
- `typedef mapped_type_allocator::const_reference const_mapped_reference`
- `typedef value_type_allocator::const_pointer const_pointer`
- `typedef value_type_allocator::const_reference const_reference`
- `typedef mapped_type_allocator::pointer mapped_pointer`
- `typedef mapped_type_allocator::reference mapped_reference`
- `typedef mapped_type_allocator::value_type mapped_type`
- `typedef Allocator::template rebind< null_mapped_type >::other mapped_type_allocator`
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef Key value_type`
- `typedef Allocator::template rebind< value_type >::other value_type_allocator`

Static Public Attributes

- static [null_mapped_type](#) `s_null_mapped`

5.198 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class
Template Reference 1257
5.197.1 Detailed Description

```
template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >
```

Specialization of value_type_base for the case where the hash value is stored alongside each value.

Definition at line 164 of file basic_types.hpp.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.198 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class
Template Reference

A concrete general-probing hash-based associative container.

Inheritance diagram for __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >:



Public Types

- typedef Allocator **allocator_type**
- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef [gp_hash_tag](#) **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**

- typedef Hash_Fn **hash_fn**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef Probe_Fn **probe_fn**
- typedef value_rebind::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- **gp_hash_table** (const hash_fn &h)
- **gp_hash_table** (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h)
- **gp_hash_table** (const [gp_hash_table](#) &other)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- **gp_hash_table** (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It >
gp_hash_table (It first, It last)

- **gp_hash_table** (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- **gp_hash_table** (const hash_fn &h, const eq_fn &e)
- **gp_hash_table** & **operator=** (const [gp_hash_table](#) &other)
- void **swap** ([gp_hash_table](#) &other)

5.198.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name detail::default_hash_fn<Key>::type, typename Eq_Fn = type-
name detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename
detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy
= typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_
Hash = detail::default_store_hash, typename Allocator = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_
Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >
```

A concrete general-probing hash-based associative container.

Definition at line 318 of file `assoc_container.hpp`.

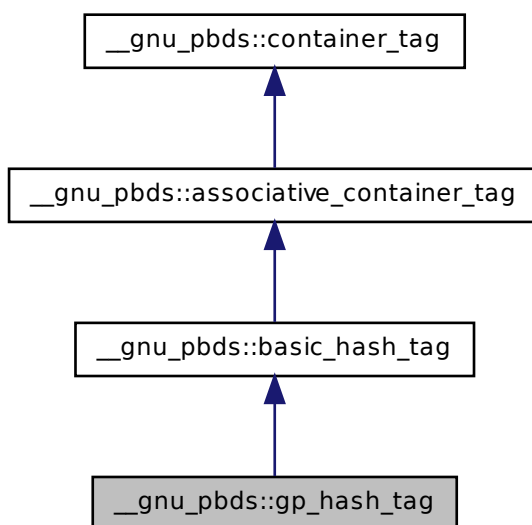
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.199 __gnu_pbds::gp_hash_tag Struct Reference

General-probing hash.

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



5.199.1 Detailed Description

General-probing hash.

Definition at line 112 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.200 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >` Class Template Reference

A list-update based associative container.

5.200 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >` Class Template Reference 1261

Inheritance diagram for `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`:



Public Types

- typedef Allocator **allocator**
- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef [list_update_tag](#) **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef Update_Policy **update_policy**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- `template<typename It >`
`list_update` (It first, It last)
- `list_update` (const [list_update](#) &other)
- `list_update` & `operator=` (const [list_update](#) &other)
- `void swap` ([list_update](#) &other)

5.200.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename
detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_
update_policy::type, class Allocator = std::allocator<char>> class __gnu_
pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >
```

A list-update based associative container.

Definition at line 654 of file `assoc_container.hpp`.

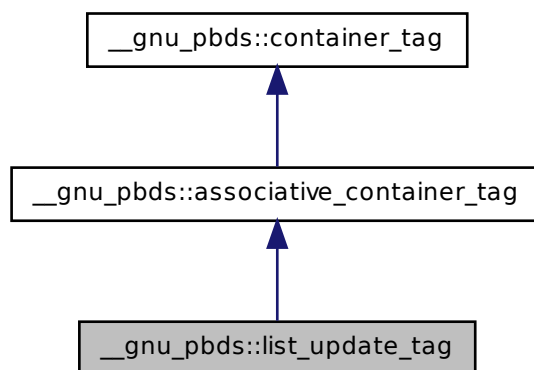
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.201 `__gnu_pbds::list_update_tag` Struct Reference

List-update.

Inheritance diagram for __gnu_pbds::list_update_tag:



5.201.1 Detailed Description

List-update.

Definition at line 136 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.202 __gnu_pbds::null_mapped_type Struct Reference

A mapped-policy indicating that an associative container is a set.

5.202.1 Detailed Description

A mapped-policy indicating that an associative container is a set.

Definition at line 89 of file `tag_and_trait.hpp`.

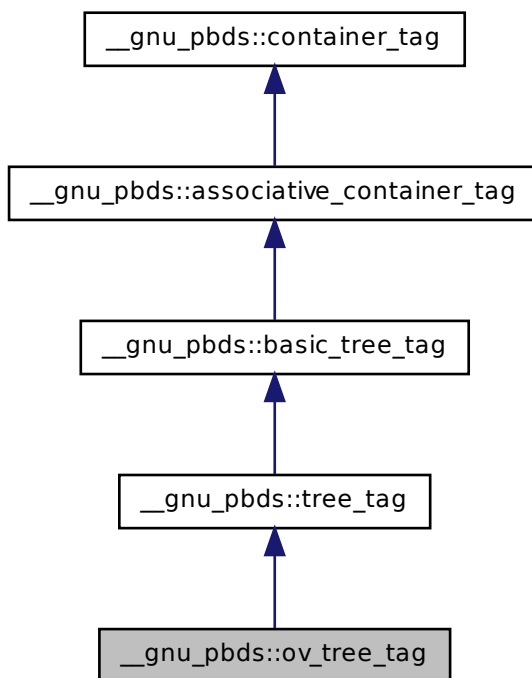
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.203 __gnu_pbds::ov_tree_tag Struct Reference

Ordered-vector tree.

Inheritance diagram for __gnu_pbds::ov_tree_tag:



5.203.1 Detailed Description

Ordered-vector tree.

Definition at line 127 of file `tag_and_trait.hpp`.

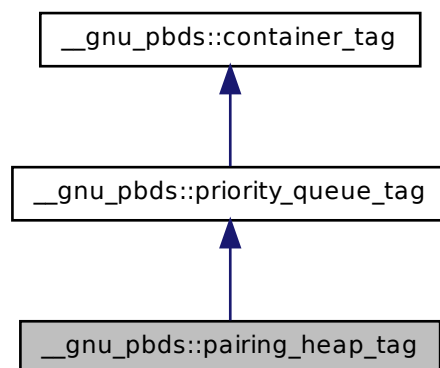
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.204 `__gnu_pbds::pairing_heap_tag` Struct Reference

Pairing-heap.

Inheritance diagram for `__gnu_pbds::pairing_heap_tag`:



5.204.1 Detailed Description

Pairing-heap.

Definition at line 142 of file `tag_and_trait.hpp`.

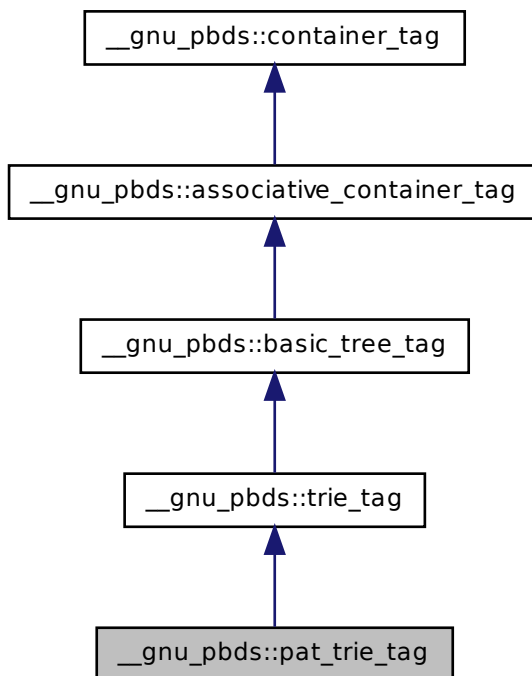
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.205 `__gnu_pbds::pat_trie_tag` Struct Reference

PATRICIA trie.

Inheritance diagram for `__gnu_pbds::pat_trie_tag`:



5.205.1 Detailed Description

PATRICIA trie.

Definition at line 133 of file `tag_and_trait.hpp`.

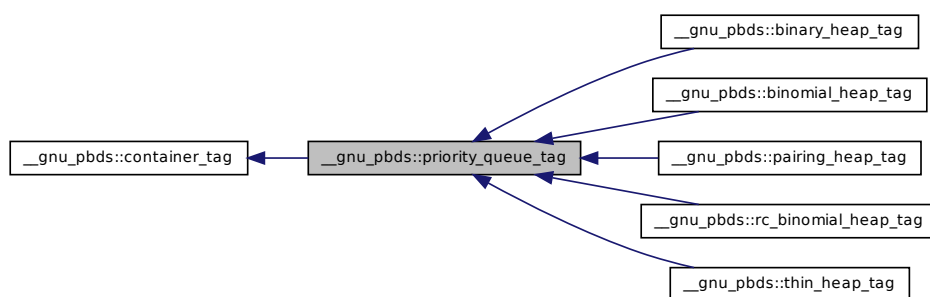
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.206 __gnu_pbds::priority_queue_tag Struct Reference

Basic priority-queue.

Inheritance diagram for __gnu_pbds::priority_queue_tag:



5.206.1 Detailed Description

Basic priority-queue.

Definition at line 139 of file `tag_and_trait.hpp`.

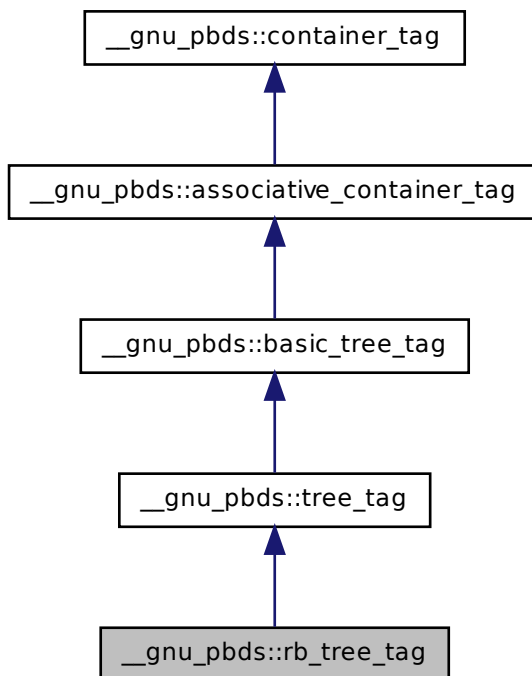
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.207 __gnu_pbds::rb_tree_tag Struct Reference

Red-black tree.

Inheritance diagram for `__gnu_pbds::rb_tree_tag`:



5.207.1 Detailed Description

Red-black tree.

Definition at line 121 of file `tag_and_trait.hpp`.

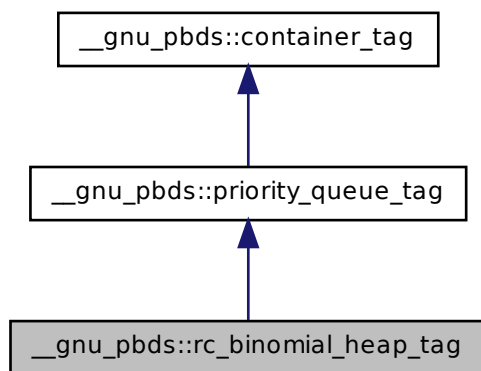
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.208 __gnu_pbds::rc_binomial_heap_tag Struct Reference

Redundant-counter binomial-heap.

Inheritance diagram for __gnu_pbds::rc_binomial_heap_tag:



5.208.1 Detailed Description

Redundant-counter binomial-heap.

Definition at line 148 of file `tag_and_trait.hpp`.

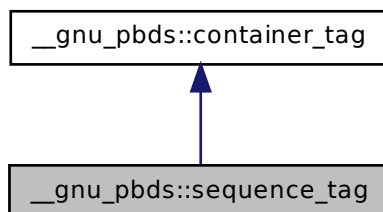
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.209 __gnu_pbds::sequence_tag Struct Reference

Basic sequence.

Inheritance diagram for `__gnu_pbds::sequence_tag`:



5.209.1 Detailed Description

Basic sequence.

Definition at line 100 of file `tag_and_trait.hpp`.

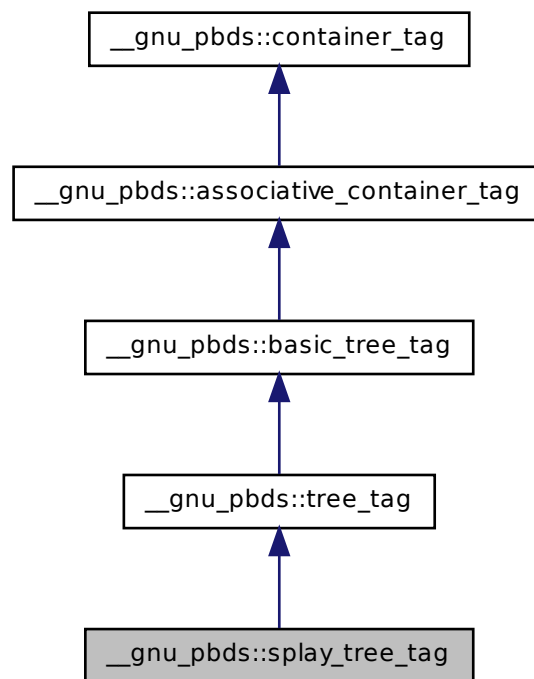
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.210 `__gnu_pbds::splay_tree_tag` Struct Reference

Splay tree.

Inheritance diagram for __gnu_pbds::splay_tree_tag:



5.210.1 Detailed Description

Splay tree.

Definition at line 124 of file `tag_and_trait.hpp`.

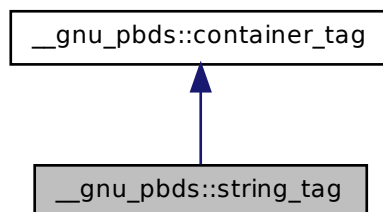
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.211 __gnu_pbds::string_tag Struct Reference

Basic string container, inclusive of strings, ropes, etc.

Inheritance diagram for `__gnu_pbds::string_tag`:



5.211.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

Definition at line 97 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

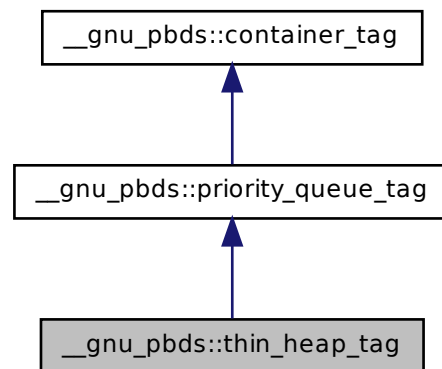
- [tag_and_trait.hpp](#)

5.212 `__gnu_pbds::thin_heap_tag` Struct Reference

Thin heap.

5.213 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >` Class Template Reference 1273

Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



5.212.1 Detailed Description

Thin heap.

Definition at line 154 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.213 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >` Class Template Reference

A concrete basic tree-based associative container.

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Tag, Allocator >::node_update **node_update**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- **tree** (const cmp_fn &c)
- template<typename It >
 tree (It first, It last, const cmp_fn &c)
- **tree** (const [tree](#) &other)
- template<typename It >
 tree (It first, It last)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

5.213.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn =
std::less<Key>, typename Tag = rb_tree_tag, template< typename Const_
Node_Iterator, typename Node_Iterator, typename Cmp_Fn_, typename
Allocator_ > class Node_Update = __gnu_pbds::null_tree_node_update, type-
name Allocator = std::allocator<char>> > class __gnu_pbds::tree< Key, Mapped,
Cmp_Fn, Tag, Node_Update, Allocator >
```

A concrete basic tree-based associative container.

Definition at line 510 of file `assoc_container.hpp`.

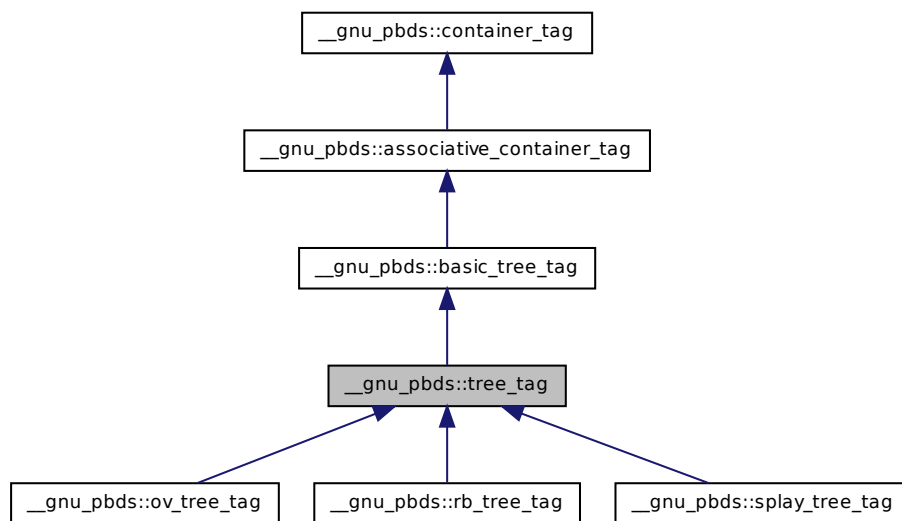
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.214 __gnu_pbds::tree_tag Struct Reference

tree.

Inheritance diagram for __gnu_pbds::tree_tag:



5.214.1 Detailed Description

tree.

Definition at line 118 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.215 `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >` Class Template Reference

A concrete basic trie-based associative container.

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef E_Access_Traits **e_access_traits**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**

- `typedef allocator_type::template rebind< mapped_type >::other mapped_rebind`
- `typedef mapped_rebind::reference mapped_reference`
- `typedef Mapped mapped_type`
- `typedef detail::trie_traits< Key, Mapped, E_Access_Traits, Node_Update, Tag, Allocator >::node_update node_update`
- `typedef base_type::point_iterator point_iterator`
- `typedef value_rebind::pointer pointer`
- `typedef value_rebind::reference reference`
- `typedef allocator_type::size_type size_type`
- `typedef allocator_type::template rebind< value_type >::other value_rebind`
- `typedef base_type::value_type value_type`

Public Member Functions

- `trie` (const `e_access_traits` &`t`)
- `template<typename It >`
`trie` (It first, It last, const `e_access_traits` &`t`)
- `trie` (const [trie](#) &`other`)
- `template<typename It >`
`trie` (It first, It last)
- `trie & operator=` (const [trie](#) &`other`)
- `void swap` ([trie](#) &`other`)

5.215.1 Detailed Description

`template<typename Key, typename Mapped, typename E_Access_Traits = typename detail::default_trie_e_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Const_Node_Iterator, typename Node_Iterator, typename E_Access_Traits_, typename Allocator_ > class Node_Update = null_trie_node_update, typename Allocator = std::allocator<char>> class __gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`

A concrete basic trie-based associative container.

Definition at line 586 of file `assoc_container.hpp`.

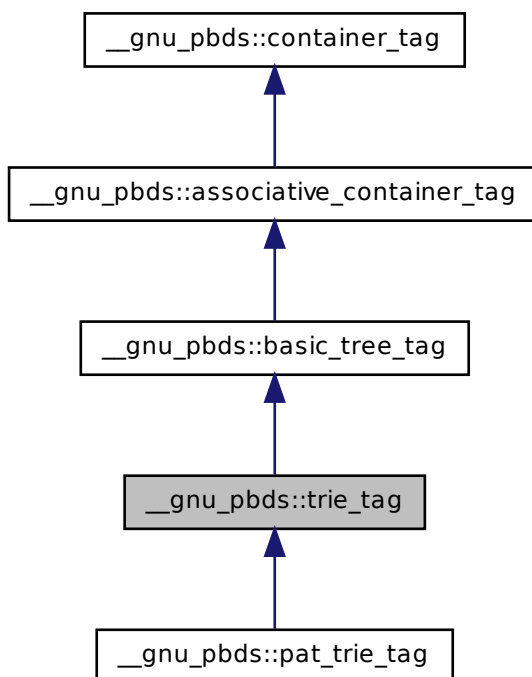
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.216 __gnu_pbds::trie_tag Struct Reference

trie.

Inheritance diagram for __gnu_pbds::trie_tag:



5.216.1 Detailed Description

trie.

Definition at line 130 of file `tag_and_trait.hpp`.

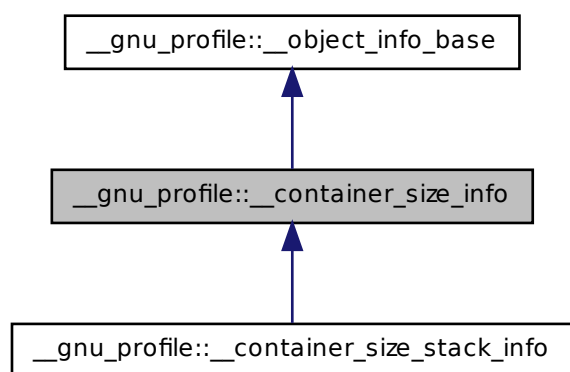
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.217 __gnu_profile::__container_size_info Class Reference

A container size instrumentation line in the object table.

Inheritance diagram for __gnu_profile::__container_size_info:



Public Member Functions

- `__container_size_info` (const `__container_size_info` &__o)
- `__container_size_info` (`__stack_t` __stack, `std::size_t` __num)
- `std::string` `__advice` () const
- void `__destruct` (`std::size_t` __num, `std::size_t` __inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__container_size_info` &__o)
- void `__resize` (`std::size_t` __from, `std::size_t` __to)
- float `__resize_cost` (`std::size_t` __from, `std::size_t`)
- `__stack_t` `__stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t` `_M_stack`

- `bool _M_valid`

5.217.1 Detailed Description

A container size instrumentation line in the object table.

Definition at line 49 of file `profiler_container_size.h`.

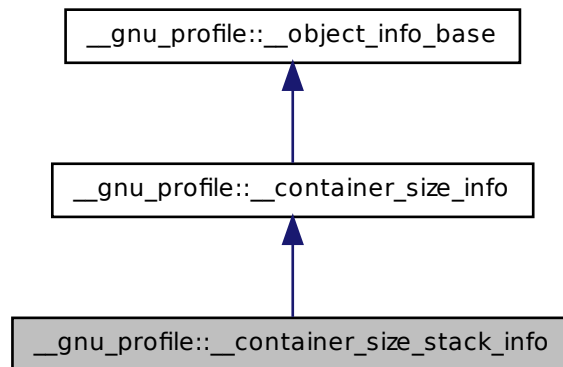
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

5.218 `__gnu_profile::__container_size_stack_info` Class Reference

A container size instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__container_size_stack_info`:



Public Member Functions

- `__container_size_stack_info` (const `__container_size_info` &`_o`)
- `std::string __advice` () const

- void `__destruct` (std::size_t __num, std::size_t __inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [__container_size_info](#) &__o)
- void `__resize` (std::size_t __from, std::size_t __to)
- float `__resize_cost` (std::size_t __from, std::size_t)
- `__stack_t` `__stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t` `_M_stack`
- bool `_M_valid`

5.218.1 Detailed Description

A container size instrumentation line in the stack table.

Definition at line 161 of file `profiler_container_size.h`.

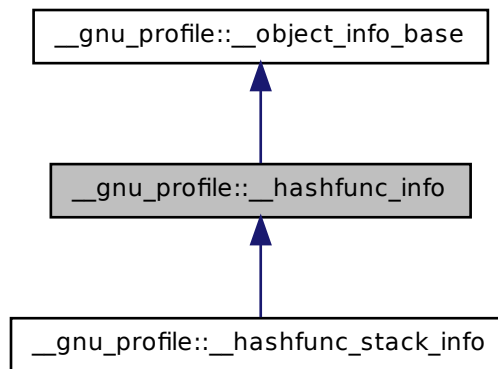
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

5.219 `__gnu_profile::__hashfunc_info` Class Reference

A hash performance instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__hashfunc_info`:



Public Member Functions

- `__hashfunc_info` (const [__hashfunc_info](#) &__o)
- `__hashfunc_info` (__stack_t __stack)
- [std::string](#) `__advice` () const
- void `__destruct` (std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [__hashfunc_info](#) &__o)
- `__stack_t` `__stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t` `_M_stack`
- bool `_M_valid`

5.219.1 Detailed Description

A hash performance instrumentation line in the object table.

Definition at line 47 of file `profiler_hash_func.h`.

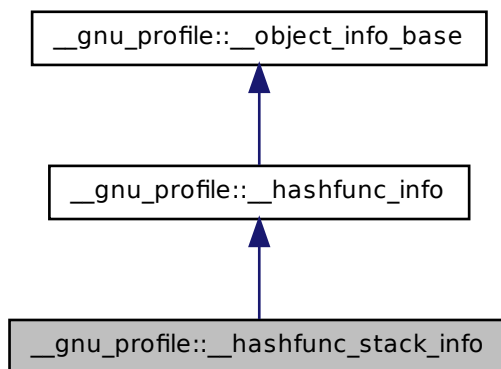
The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

5.220 `__gnu_profile::__hashfunc_stack_info` Class Reference

A hash performance instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__hashfunc_stack_info`:



Public Member Functions

- `__hashfunc_stack_info` (const `__hashfunc_info` &`__o`)
- `std::string __advice` () const
- void `__destruct` (std::size_t `__chain`, std::size_t `__accesses`, std::size_t `__hops`)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__hashfunc_info` &`__o`)
- `__stack_t __stack` () const
- void `__write` (FILE *`__f`) const

Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

5.220.1 Detailed Description

A hash performance instrumentation line in the stack table.

Definition at line 102 of file `profiler_hash_func.h`.

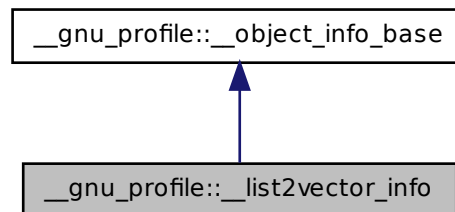
The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

5.221 `__gnu_profile::__list2vector_info` Class Reference

A list-to-vector instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__list2vector_info`:



Public Member Functions

- `__list2vector_info (__stack_t __stack)`
- `__list2vector_info (const __list2vector_info &__o)`
- `std::string __advice () const`
- `bool __is_valid () const`

- `bool __is_valid ()`
- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __list2vector_info &__o)`
- `void __opr_insert (std::size_t __shift, std::size_t __size)`
- `void __opr_iterate (std::size_t __num)`
- `std::size_t __resize ()`
- `void __resize (std::size_t __from, std::size_t)`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

Protected Attributes

- `__stack_t _M_stack`

5.221.1 Detailed Description

A list-to-vector instrumentation line in the object table.

Definition at line 49 of file `profiler_list_to_vector.h`.

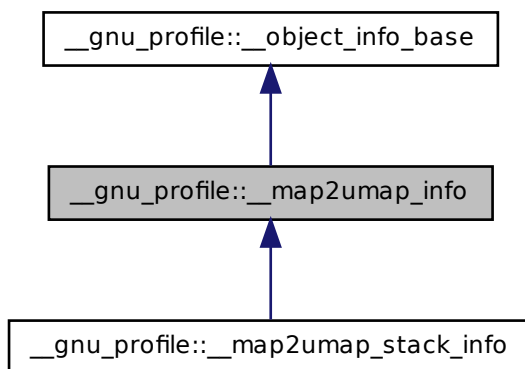
The documentation for this class was generated from the following file:

- [profiler_list_to_vector.h](#)

5.222 `__gnu_profile::__map2umap_info` Class Reference

A map-to-unordered_map instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__map2umap_info`:



Public Member Functions

- `__map2umap_info` (`__stack_t __stack`)
- `__map2umap_info` (`const __map2umap_info &__o`)
- `std::string __advice` () `const`
- `bool __is_valid` () `const`
- `float __magnitude` () `const`
- `void __merge` (`const __map2umap_info &__o`)
- `void __record_erase` (`std::size_t __size`, `std::size_t __count`)
- `void __record_find` (`std::size_t __size`)
- `void __record_insert` (`std::size_t __size`, `std::size_t __count`)
- `void __record_invalidate` ()
- `void __record_iterate` (`std::size_t __count`)
- `__stack_t __stack` () `const`
- `void __write` (`FILE *__f`) `const`

Protected Attributes

- `__stack_t _M_stack`

5.222.1 Detailed Description

A map-to-unordered_map instrumentation line in the object table.

Definition at line 73 of file `profiler_map_to_unordered_map.h`.

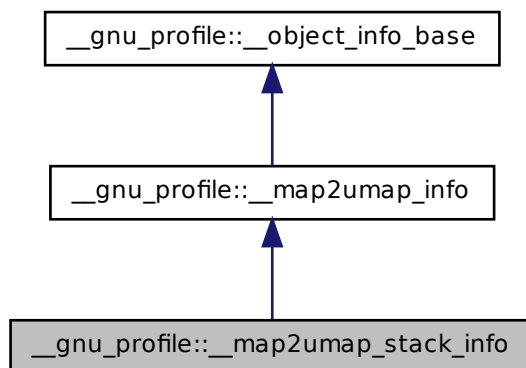
The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

5.223 `__gnu_profile::__map2umap_stack_info` Class Reference

A map-to-unordered_map instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__map2umap_stack_info`:



Public Member Functions

- `__map2umap_stack_info` (const `__map2umap_info` &`__o`)
- `std::string __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__map2umap_info` &`__o`)

- void **__record_erase** (std::size_t __size, std::size_t __count)
- void **__record_find** (std::size_t __size)
- void **__record_insert** (std::size_t __size, std::size_t __count)
- void **__record_invalidate** ()
- void **__record_iterate** (std::size_t __count)
- __stack_t **__stack** () const
- void **__write** (FILE *__f) const

Protected Attributes

- __stack_t **_M_stack**

5.223.1 Detailed Description

A map-to-unordered_map instrumentation line in the stack table.

Definition at line 177 of file profiler_map_to_unordered_map.h.

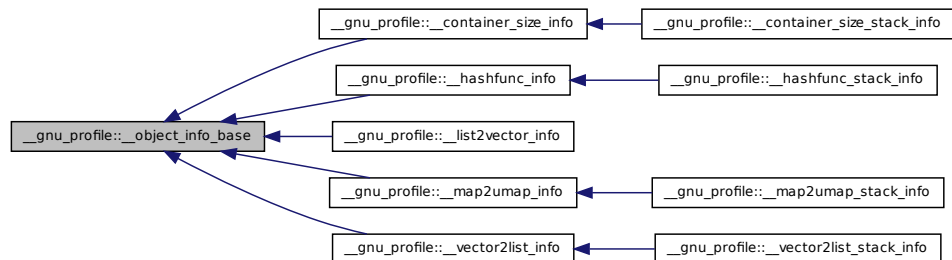
The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

5.224 __gnu_profile::__object_info_base Class Reference

Base class for a line in the object table.

Inheritance diagram for __gnu_profile::__object_info_base:



Public Member Functions

- `__object_info_base` (`__stack_t __stack`)
- `__object_info_base` (`const __object_info_base &__o`)
- `bool __is_valid` () `const`
- `__stack_t __stack` () `const`
- `virtual void __write` (`FILE *f`) `const` =0

Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

5.224.1 Detailed Description

Base class for a line in the object table.

Definition at line 130 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

5.225 `__gnu_profile::__reentrance_guard` Struct Reference

Reentrance guard.

Static Public Member Functions

- `static bool __get_in` ()
- `static bool & __inside` ()

5.225.1 Detailed Description

Reentrance guard. Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 65 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

5.226 `__gnu_profile::__stack_hash` Class Reference

Hash function for summary trace using call stack as index.

Public Member Functions

- `std::size_t operator() (__stack_t __s) const`
- `bool operator() (__stack_t __stack1, __stack_t __stack2) const`

5.226.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 96 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

5.227 `__gnu_profile::__stack_info_base< __object_info >` Class Template Reference

Base class for a line in the stack table.

Public Member Functions

- `__stack_info_base (const __object_info &__info)=0`
- `virtual const char * __get_id () const =0`
- `virtual float __magnitude () const =0`
- `void __merge (const __object_info &__info)=0`

5.227.1 Detailed Description

```
template<typename __object_info> class __gnu_profile::__stack_info_base< __-
object_info >
```

Base class for a line in the stack table.

Definition at line 161 of file `profiler_node.h`.

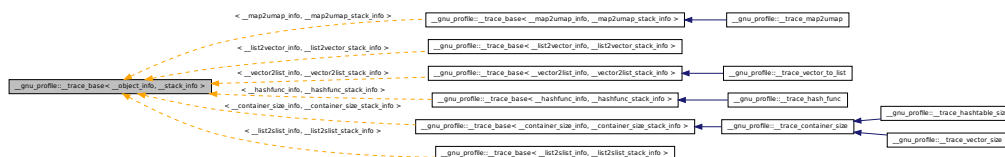
The documentation for this class was generated from the following file:

- [profiler_node.h](#)

5.228 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

Base class for all trace producers.

Inheritance diagram for `__gnu_profile::__trace_base< __object_info, __stack_info >`:



Public Member Functions

- void `__add_object` (`__object_t` object, `__object_info` __info)
- void `__collect_warnings` (`__warning_vector_t` &__warnings)
- `__object_info` * `__get_object_info` (`__object_t` __object)
- void `__retire_object` (`__object_t` __object)
- void `__write` (`FILE` * __f)

Protected Attributes

- const char * `__id`

5.228.1 Detailed Description

`template<typename __object_info, typename __stack_info> class __gnu_profile::__trace_base< __object_info, __stack_info >`

Base class for all trace producers.

Definition at line 190 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler_trace.h](#)

5.229 `__gnu_profile::__trace_container_size` Class Reference

Container size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_container_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, `__container_size_info` info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, `std::size_t` __inum)
- void **__destruct** (const void *__obj, `std::size_t` __num, `std::size_t` __inum)
- `__container_size_info` * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, `std::size_t` __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.229.1 Detailed Description

Container size instrumentation trace producer.

Definition at line 171 of file `profiler_container_size.h`.

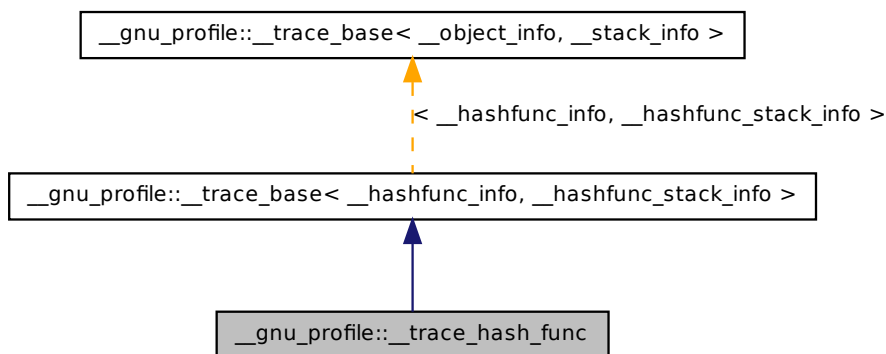
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

5.230 `__gnu_profile::__trace_hash_func` Class Reference

Hash performance instrumentation producer.

Inheritance diagram for __gnu_profile::__trace_hash_func:



Public Member Functions

- void **__add_object** (__object_t object, __hashfunc_info __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void *__obj, std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- [__hashfunc_info](#) * **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.230.1 Detailed Description

Hash performance instrumentation producer.

Definition at line 112 of file profiler_hash_func.h.

The documentation for this class was generated from the following file:

- profiler_hash_func.h

5.231 `__gnu_profile::__trace_hashtable_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, `__container_size_info` info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, `std::size_t` __inum)
- void **__destruct** (const void *__obj, `std::size_t` __num, `std::size_t` __inum)
- `__container_size_info` * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, `std::size_t` __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.231.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 49 of file `profiler_hashtable_size.h`.

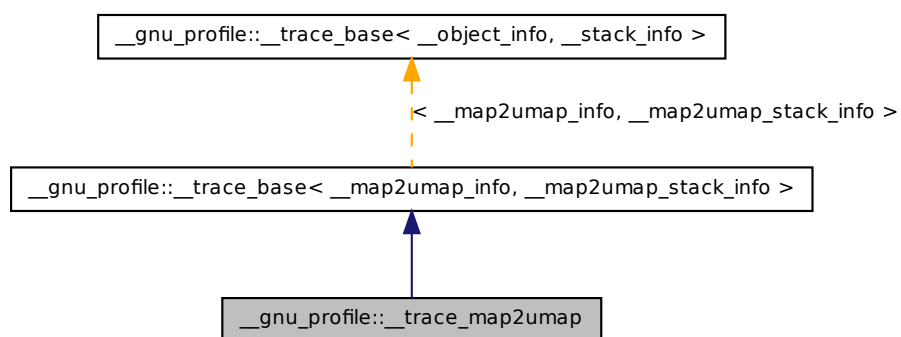
The documentation for this class was generated from the following file:

- [profiler_hashtable_size.h](#)

5.232 `__gnu_profile::__trace_map2umap` Class Reference

Map-to-unordered_map instrumentation producer.

Inheritance diagram for __gnu_profile::__trace_map2umap:



Public Member Functions

- void **__add_object** (__object_t object, __map2umap_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- [__map2umap_info](#) * **__get_object_info** (__object_t __object)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.232.1 Detailed Description

Map-to-unordered_map instrumentation producer.

Definition at line 186 of file profiler_map_to_unordered_map.h.

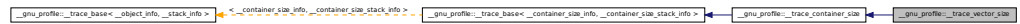
The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

5.233 `__gnu_profile::__trace_vector_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, `__container_size_info` __info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, `std::size_t` __inum)
- void **__destruct** (const void *__obj, `std::size_t` __num, `std::size_t` __inum)
- `__container_size_info` * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, `std::size_t` __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.233.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 49 of file `profiler_vector_size.h`.

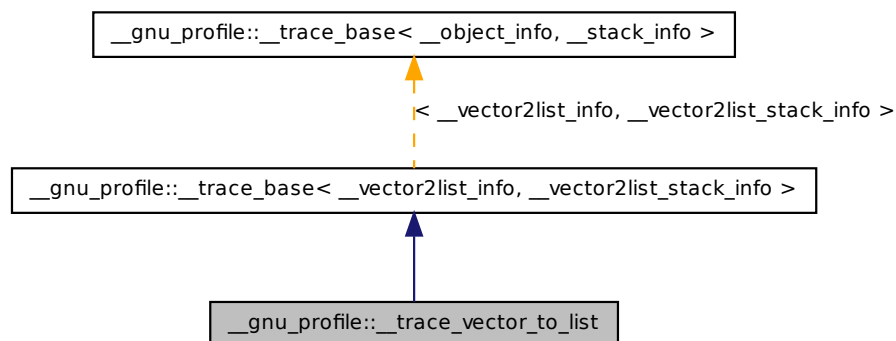
The documentation for this class was generated from the following file:

- [profiler_vector_size.h](#)

5.234 `__gnu_profile::__trace_vector_to_list` Class Reference

Vector-to-list instrumentation producer.

Inheritance diagram for __gnu_profile::__trace_vector_to_list:



Public Member Functions

- void **__add_object** (__object_t object, __vector2list_info __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void *__obj)
- [__vector2list_info](#) * **__find** (const void *__obj)
- [__vector2list_info](#) * **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__invalid_operator** (const void *__obj)
- float **__list_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__opr_find** (const void *__obj, std::size_t __size)
- void **__opr_insert** (const void *__obj, std::size_t __pos, std::size_t __num)
- void **__opr_iterate** (const void *__obj, std::size_t __num)
- void **__resize** (const void *__obj, std::size_t __from, std::size_t __to)
- void **__retire_object** (__object_t __object)
- float **__vector_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.234.1 Detailed Description

Vector-to-list instrumentation producer.

Definition at line 165 of file profiler_vector_to_list.h.

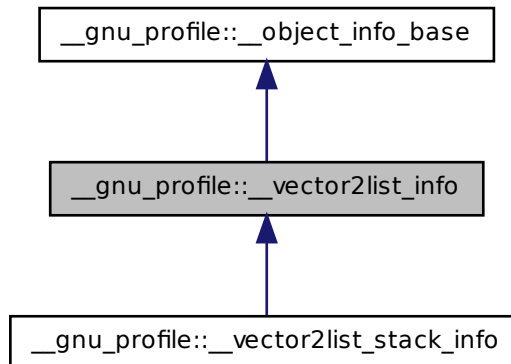
The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

5.235 `__gnu_profile::__vector2list_info` Class Reference

A vector-to-list instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__vector2list_info`:



Public Member Functions

- `__vector2list_info` (`__stack_t` `__stack`)
- `__vector2list_info` (`const` `__vector2list_info` &`__o`)
- `std::string` `__advice` () `const`
- `bool` `__is_valid` () `const`
- `bool` `__is_valid` ()

- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __vector2list_info &__o)`
- `void __opr_find (std::size_t __size)`
- `void __opr_insert (std::size_t __pos, std::size_t __num)`
- `void __opr_iterate (std::size_t __num)`
- `void __resize (std::size_t __from, std::size_t)`
- `std::size_t __resize ()`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

Protected Attributes

- `__stack_t _M_stack`

5.235.1 Detailed Description

A vector-to-list instrumentation line in the object table.

Definition at line 47 of file `profiler_vector_to_list.h`.

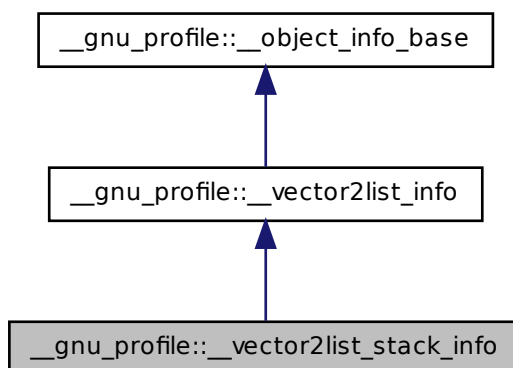
The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

5.236 `__gnu_profile::__vector2list_stack_info` Class Reference

A vector-to-list instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__vector2list_stack_info`:



Public Member Functions

- `__vector2list_stack_info` (const `__vector2list_info` &__o)
- `std::string` `__advice` () const
- `bool` `__is_valid` ()
- `bool` `__is_valid` () const
- `std::size_t` `__iterate` ()
- `float` `__list_cost` ()
- `float` `__magnitude` () const
- `void` `__merge` (const `__vector2list_info` &__o)
- `void` `__opr_find` (std::size_t __size)
- `void` `__opr_insert` (std::size_t __pos, std::size_t __num)
- `void` `__opr_iterate` (std::size_t __num)
- `void` `__resize` (std::size_t __from, std::size_t)
- `std::size_t` `__resize` ()
- `void` `__set_invalid` ()
- `void` `__set_list_cost` (float __lc)
- `void` `__set_vector_cost` (float __vc)
- `std::size_t` `__shift_count` ()
- `__stack_t` `__stack` () const
- `void` `__write` (FILE *__f) const

Protected Attributes

- `__stack_t __M_stack`

5.236.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 155 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

5.237 `__gnu_profile::__warning_data` Struct Reference

Representation of a warning.

Public Member Functions

- `__warning_data` (float `__m`, `__stack_t` `__c`, const char *`__id`, const [std::string](#) &`__msg`)
- bool `operator<` (const [__warning_data](#) &`__other`) const

Public Attributes

- `__stack_t __context`
- float `__magnitude`
- const char * `__warning_id`
- [std::string](#) `__warning_message`

5.237.1 Detailed Description

Representation of a warning.

Definition at line 80 of file `profiler_trace.h`.

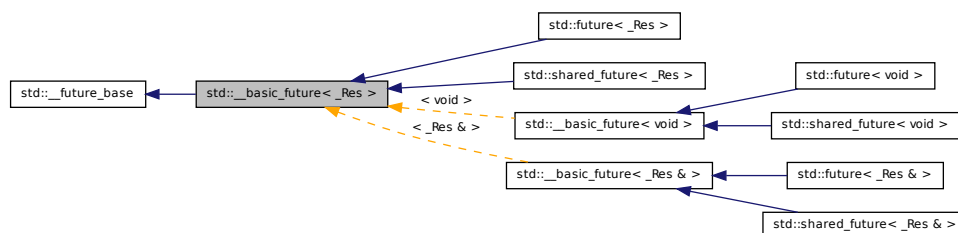
The documentation for this struct was generated from the following file:

- [profiler_trace.h](#)

5.238 std::__basic_future< _Res > Class Template Reference

Common implementation for future and [shared_future](#).

Inheritance diagram for std::__basic_future< _Res >:



Public Member Functions

- `__basic_future` (const `__basic_future` &)
- `__basic_future` & `operator=` (const `__basic_future` &)
- `bool valid ()` const
- `void wait ()` const
- `template<typename _Rep, typename _Period >`
`bool wait_for` (const `chrono::duration< _Rep, _Period >` &__rel) const
- `template<typename _Clock, typename _Duration >`
`bool wait_until` (const `chrono::time_point< _Clock, _Duration >` &__abs)
const

Protected Types

- `typedef __future_base::__Result< _Res > & __result_type`
- `typedef shared_ptr< _State > __state_type`

Protected Member Functions

- `__basic_future` (const `__state_type` &__state)
- `__basic_future` (const `shared_future< _Res >` &)
- `__basic_future` (`shared_future< _Res >` &&)
- `__basic_future` (`future< _Res >` &&)

5.239 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference 1303

- [__result_type __M_get_result\(\)](#)
- [void __M_swap\(__basic_future & __that\)](#)

5.238.1 Detailed Description

template<typename _Res> class std::__basic_future< _Res >

Common implementation for future and [shared_future](#).

Definition at line 480 of file future.

5.238.2 Member Function Documentation

5.238.2.1 **template<typename _Res> __result_type std::__basic_future< _Res >::__M_get_result() [inline, protected]**

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

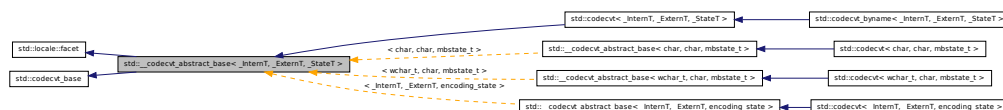
The documentation for this class was generated from the following file:

- [future](#)

5.239 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference

Common base for codecvt functions.

Inheritance diagram for `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Protected Member Functions

- **__codecvt_abstract_base** (size_t __refs=0)
- virtual bool **do_always_noconv** () const =0 throw ()
- virtual int **do_encoding** () const =0 throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_max_length** () const =0 throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.239.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT> class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >

Common base for codecvt functions. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 67 of file `codecvt.h`.

5.239.2 Member Function Documentation

5.239.2.1 **template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected, pure virtual]**

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implemented in `std::codecvt< _InternT, _ExternT, _StateT >`.

```

5.239.2.2 template<typename _InternT, typename _ExternT, typename
_StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
_StateT >::in ( state_type & __state, const extern_type * __from,
const extern_type * __from_end, const extern_type * & __from_next,
intern_type * __to, intern_type * __to_end, intern_type * &
__to_next ) const [inline]

```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.239.2.3 `template<typename _InternT, typename _ExternT, typename
_StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
_StateT >::out (state_type & __state, const intern_type * __from,
const intern_type * __from_end, const intern_type *& __from_next,
extern_type * __to, extern_type * __to_end, extern_type *&
__to_next) const [inline]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

```

5.239.2.4 template<typename _InternT, typename _ExternT, typename
             _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
             _StateT >::unshift ( state_type & __state, extern_type * __to,
             extern_type * __to_end, extern_type *& __to_next ) const
             [inline]

```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to [in\(\)](#) had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

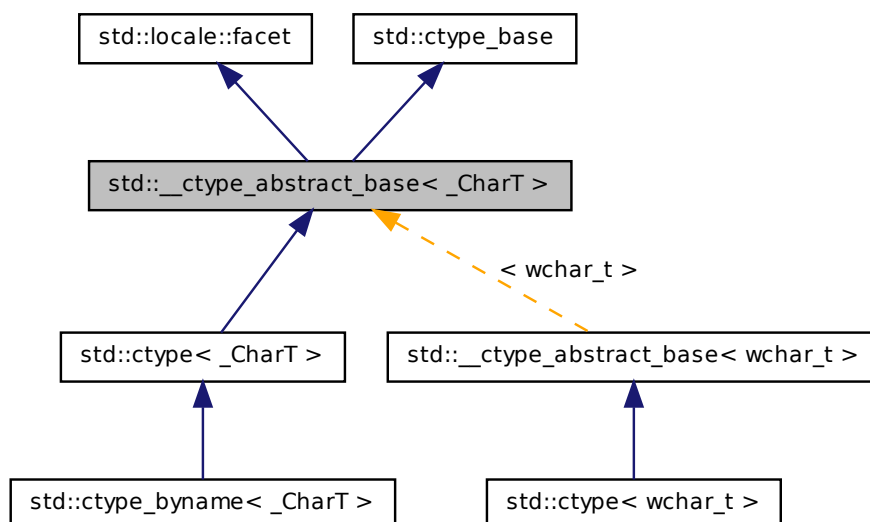
The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.240 std::__ctype_abstract_base< _CharT > Class Template Reference

Common base for ctype facet.

Inheritance diagram for `std::__ctype_abstract_base<_CharT>`:



Public Types

- typedef const int * **__to_type**
- typedef `_CharT` **char_type**
- typedef unsigned short **mask**

Public Member Functions

- bool **is** (mask `__m`, **char_type** `__c`) const
- const **char_type** * **is** (const **char_type** * `__lo`, const **char_type** * `__hi`, mask * `__vec`) const
- char **narrow** (**char_type** `__c`, char `__default`) const
- const **char_type** * **narrow** (const **char_type** * `__lo`, const **char_type** * `__hi`, char `__default`, char * `__to`) const
- const **char_type** * **scan_is** (mask `__m`, const **char_type** * `__lo`, const **char_type** * `__hi`) const

- const [char_type](#) * [scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- [__ctype_abstract_base](#) (size_t __refs=0)
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const =0
- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const =0
- virtual char [do_narrow](#) ([char_type](#), char __dfault) const =0
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __dfault, char * __dest) const =0
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const =0
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const =0
- virtual [char_type](#) [do_tolower](#) ([char_type](#)) const =0
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const =0
- virtual [char_type](#) [do_toupper](#) ([char_type](#)) const =0
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const =0

- virtual const char * `do_widen` (const char *__lo, const char *__hi, `char_type` *__dest) const =0
- virtual `char_type do_widen` (char) const =0

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (__c_locale &__cloc) throw ()
- static void `_S_create_c_locale` (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void `_S_destroy_c_locale` (__c_locale &__cloc)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char * `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (__c_locale __cloc, const char *__s)

Friends

- class `locale::_Impl`

5.240.1 Detailed Description

`template<typename _CharT> class std::__ctype_abstract_base<_CharT>`

Common base for ctype facet. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file `locale_facets.h`.

5.240.2 Member Typedef Documentation

5.240.2.1 `template<typename _CharT> typedef _CharT
std::__ctype_abstract_base<_CharT>::char_type`

Typedef for the template parameter.

Reimplemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Definition at line 148 of file `locale_facets.h`.

5.240.3 Member Function Documentation

5.240.3.1 `template<typename _CharT> virtual bool std::__ctype_abstract_base<_CharT>::do_is (mask __m, char_type __c) const [protected, pure virtual]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

c The `char_type` to find the mask of.

m The mask to compare against.

Returns

$(M \& m) \neq 0$.

Implemented in `std::ctype<_CharT>`.

5.240.3.2 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const [protected, pure virtual]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Implemented in `std::ctype<_CharT>`.

5.240.3.3 `template<typename _CharT> virtual char std::__ctype_abstract_base<_CharT>::do_narrow (char_type, char __dfault) const [protected, pure virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

Returns

- The converted `char`.

Implemented in `std::ctype<_CharT>`.

5.240.3.4 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, pure virtual]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Implemented in [std::ctype<_CharT>](#).

5.240.3.5 `template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_scan_is (mask
__m, const char_type * __lo, const char_type * __hi) const
[protected, pure virtual]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

Implemented in [std::ctype<_CharT>](#).

5.240.3.6 `template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_scan_not (mask
__m, const char_type * __lo, const char_type * __hi) const
[protected, pure virtual]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implemented in `std::ctype<_CharT>`.

```
5.240.3.7 template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_tolower ( char_type )
const [protected, pure virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

Implemented in `std::ctype<_CharT>`.

```
5.240.3.8 template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_tolower ( char_type
* __lo, const char_type * __hi ) const [protected, pure
virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implemented in [std::ctype<_CharT>](#).

5.240.3.9 `template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_toupper (char_type)
const [protected, pure virtual]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Implemented in [std::ctype<_CharT>](#).

5.240.3.10 `template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_toupper (char_type
* __lo, const char_type * __hi) const [protected, pure
virtual]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns*hi*.Implemented in `std::ctype<_CharT>`.

5.240.3.11 `template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_widen (char) const
[protected, pure virtual]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters*c* The char to convert.**Returns**

The converted char_type

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

5.240.3.12 `template<typename _CharT> virtual const char*
std::__ctype_abstract_base<_CharT>::do_widen (const
char * __lo, const char * __hi, char_type * __dest) const
[protected, pure virtual]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters*lo* Pointer to start range.*hi* Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Implemented in [std::ctype<_CharT>](#).

5.240.3.13 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::is (const char_type *
__lo, const char_type * __hi, mask * __vec) const [inline]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of [ctype<char_type>::do_is\(\)](#).

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.240.3.14 `template<typename _CharT> bool std::__ctype_abstract_base<
_CharT>::is (mask __m, char_type __c) const [inline]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m. It does so by returning the value of [ctype<char_type>::do_is\(\)](#).

Parameters

c The char_type to compare the mask of.

m The mask to compare against.

Returns

(M & m) != 0.

5.240 std::__ctype_abstract_base<_CharT> Class Template Reference 1319

Definition at line 161 of file locale_facets.h.

Referenced by std::regex_traits<_Ch_type>::isctype(), and std::basic_istream<_CharT, _Traits>::sentry::sentry().

5.240.3.15 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::narrow (const char_type
* __lo, const char_type * __hi, char __dfault, char * __to) const
[inline]`

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

5.240.3.16 `template<typename _CharT> char std::__ctype_abstract_base<
_CharT>::narrow (char_type __c, char __dfault) const
[inline]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 323 of file locale_facets.h.

Referenced by `std::time_put<_CharT, _OutIter >::put()`.

5.240.3.17 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT >::scan_is (mask __m,
const char_type * __lo, const char_type * __hi) const [inline]`

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning `ctype<char_type>::do_scan_is()`.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

5.240.3.18 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT >::scan_not (mask __m,
const char_type * __lo, const char_type * __hi) const [inline]`

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning `ctype<char_type>::do_scan_not()`.

Parameters

m The mask to compare against.

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.240.3.19 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c) const [inline]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper(c).

Parameters

c The char_type to convert.

Returns

The lowercase char_type if convertible, else *c*.

Definition at line 253 of file locale_facets.h.

5.240.3.20 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower (char_type * __lo, const char_type * __hi) const [inline]`

Convert array to lowercase.

This function converts each char_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file locale_facets.h.

5.240.3.21 `template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::toupper (char_type __c) const [inline]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.240.3.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::toupper (char_type * __lo, const char_type * __hi) const [inline]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.240.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::widen (char __c) const [inline]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted `char_type`.

Definition at line 285 of file `locale_facets.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::regex_traits<_Char_type>::isctype()`, and `std::operator<<()`.

5.240.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const [inline]`

Widen array to `char_type`.

This function converts each `char` in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file `locale_facets.h`.

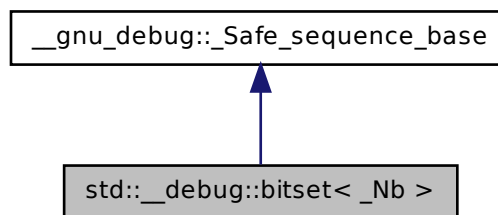
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.241 `std::__debug::bitset< _Nb >` Class Template Reference

Class `std::bitset` with additional safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::bitset< _Nb >`:



Public Member Functions

- **bitset** (unsigned long long __val)
- template<class _CharT, class _Traits, class _Alloc >
bitset (const `std::basic_string`< _CharT, _Traits, _Alloc > &__str, type-name `std::basic_string`< _CharT, _Traits, _Alloc >::size_type __pos, type-name `std::basic_string`< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))
- **bitset** (const `_Base` &__x)
- template<typename _CharT, typename _Traits, typename _Alloc >
bitset (const `std::basic_string`< _CharT, _Traits, _Alloc > &__str, type-name `std::basic_string`< _CharT, _Traits, _Alloc >::size_type __pos=0, type-name `std::basic_string`< _CharT, _Traits, _Alloc >::size_type __n=(`std::basic_string`< _CharT, _Traits, _Alloc >::npos))
- **bitset** (const char *__str)
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_invalidate_all` () const
- `bitset`< _Nb > & **flip** ()
- `bitset`< _Nb > & **flip** (size_t __pos)
- bool **operator!=** (const `bitset`< _Nb > &__rhs) const

- [bitset](#)<_Nb> & **operator**&= (const [bitset](#)<_Nb> &__rhs)
- [bitset](#)<_Nb> **operator**<< (size_t __pos) const
- [bitset](#)<_Nb> & **operator**<<= (size_t __pos)
- bool **operator**== (const [bitset](#)<_Nb> &__rhs) const
- [bitset](#)<_Nb> **operator**>> (size_t __pos) const
- [bitset](#)<_Nb> & **operator**>>= (size_t __pos)
- reference **operator**[] (size_t __pos)
- bool **operator**[] (size_t __pos) const
- [bitset](#)<_Nb> & **operator**^= (const [bitset](#)<_Nb> &__rhs)
- [bitset](#)<_Nb> & **operator**|= (const [bitset](#)<_Nb> &__rhs)
- [bitset](#)<_Nb> **operator**~ () const
- [bitset](#)<_Nb> & **reset** ()
- [bitset](#)<_Nb> & **reset** (size_t __pos)
- [bitset](#)<_Nb> & **set** ()
- [bitset](#)<_Nb> & **set** (size_t __pos, bool __val=true)
- [std::basic_string](#)< char, [std::char_traits](#)< char >, [std::allocator](#)< char > > **to_string** (char __zero, char __one= '1') const
- template<typename _CharT >
[std::basic_string](#)< _CharT, [std::char_traits](#)< _CharT >, [std::allocator](#)< _CharT > > **to_string** () const
- template<typename _CharT, typename _Traits >
[std::basic_string](#)< _CharT, _Traits, [std::allocator](#)< _CharT > > **to_string** () const
- template<class _CharT >
[std::basic_string](#)< _CharT, [std::char_traits](#)< _CharT >, [std::allocator](#)< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- [std::basic_string](#)< char, [std::char_traits](#)< char >, [std::allocator](#)< char > > **to_string** () const
- template<class _CharT, class _Traits >
[std::basic_string](#)< _CharT, _Traits, [std::allocator](#)< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits, class _Alloc >
[std::basic_string](#)< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<typename _CharT, typename _Traits, typename _Alloc >
[std::basic_string](#)< _CharT, _Traits, _Alloc > **to_string** () const

Public Attributes

- _Safe_iterator_base * [_M_const_iterators](#)
- _Safe_iterator_base * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.241.1 Detailed Description

`template<size_t _Nb> class std::__debug::bitset< _Nb >`

Class [std::bitset](#) with additional safety/checking/debug instrumentation.

Definition at line 43 of file `debug/bitset`.

5.241.2 Member Function Documentation

5.241.2.1 void `__gnu_debug::_Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.241.2.2 void `__gnu_debug::_Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.241.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw ()` [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.241.2.4 void __gnu_debug::Safe_sequence_base::M_invalidate_all ()
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe_base.h.

5.241.2.5 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.241.2.6 void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.241.3 Member Data Documentation

5.241.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.241.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.241.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

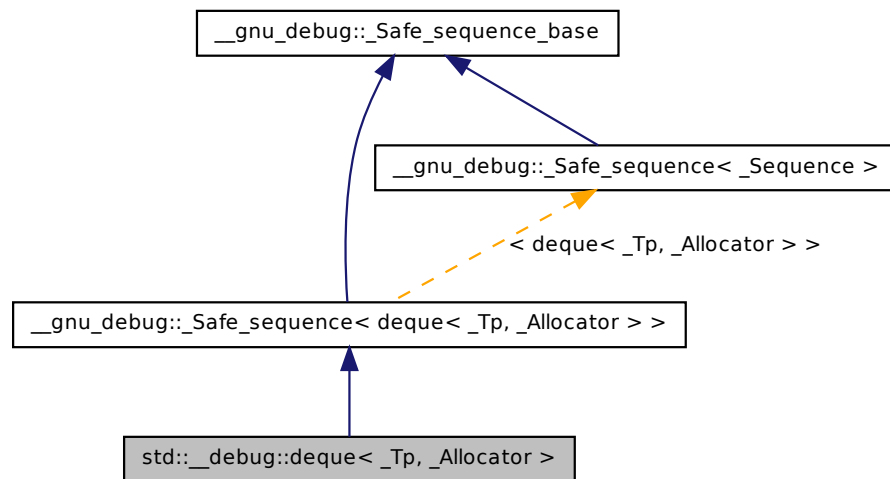
The documentation for this class was generated from the following file:

- [debug/bitset](#)

5.242 std::__debug::deque< _Tp, _Allocator > Class Template Reference

Class [std::deque](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::deque< _Tp, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**

- typedef [__gnu_debug::__Safe_iterator](#)< typename [_Base::const_iterator](#), [deque](#) > **const_iterator**
- typedef [_Base::const_pointer](#) **const_pointer**
- typedef [_Base::const_reference](#) **const_reference**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef [_Base::difference_type](#) **difference_type**
- typedef [__gnu_debug::__Safe_iterator](#)< typename [_Base::iterator](#), [deque](#) > **iterator**
- typedef [_Base::pointer](#) **pointer**
- typedef [_Base::reference](#) **reference**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef [_Base::size_type](#) **size_type**
- typedef [_Tp](#) **value_type**

Public Member Functions

- **deque** (const [_Allocator](#) &__a=_Allocator())
- **deque** (size_type __n)
- template<class [_InputIterator](#) >
deque ([_InputIterator](#) __first, [_InputIterator](#) __last, const [_Allocator](#) &__a=_Allocator())
- **deque** ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- **deque** (const [deque](#) &__x)
- **deque** (size_type __n, const [_Tp](#) &__value, const [_Allocator](#) &__a=_Allocator())
- **deque** (const [_Base](#) &__x)
- **deque** ([deque](#) &&__x)
- [_Base](#) & **_M_base** ()
- const [_Base](#) & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** ([_Predicate](#) __pred)
- void **_M_transfer_iter** (const [_Safe_iterator](#)< [_Iterator](#), [deque](#)< [_Tp](#), [_Allocator](#) > > &__x)
- template<class [_InputIterator](#) >
void **assign** ([_InputIterator](#) __first, [_InputIterator](#) __last)
- void **assign** (size_type __n, const [_Tp](#) &__t)
- void **assign** ([initializer_list](#)< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- [const_iterator](#) **cbegin** () const

- [const_iterator](#) **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- template<typename... _Args>
[iterator](#) **emplace** ([iterator](#) __position, _Args &&...__args)
- template<typename... _Args>
void **emplace_back** (_Args &&...__args)
- template<typename... _Args>
void **emplace_front** (_Args &&...__args)
- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- [iterator](#) **erase** ([iterator](#) __first, [iterator](#) __last)
- [iterator](#) **erase** ([iterator](#) __position)
- reference **front** ()
- const_reference **front** () const
- [iterator](#) **insert** ([iterator](#) __position, const _Tp &__x)
- [iterator](#) **insert** ([iterator](#) __position, _Tp &&__x)
- void **insert** ([iterator](#) __p, [initializer_list](#)< value_type > __l)
- void **insert** ([iterator](#) __position, size_type __n, const _Tp &__x)
- template<class _InputIterator >
void **insert** ([iterator](#) __position, _InputIterator __first, _InputIterator __last)
- [deque](#) & **operator=** ([deque](#) &&__x)
- [deque](#) & **operator=** (const [deque](#) &__x)
- [deque](#) & **operator=** ([initializer_list](#)< value_type > __l)
- const_reference **operator[]** (size_type __n) const
- reference **operator[]** (size_type __n)
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_Tp &&__x)
- void **push_back** (const _Tp &__x)
- void **push_front** (const _Tp &__x)
- void **push_front** (_Tp &&__x)
- [reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rend** () const
- [reverse_iterator](#) **rend** ()
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **swap** ([deque](#) &__x)

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.242.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::deque<_Tp, _Allocator>

Class [std::deque](#) with safety/checking/debug instrumentation.

Definition at line 43 of file debug/deque.

5.242.2 Member Function Documentation

5.242.2.1 void [__gnu_debug::Safe_sequence_base::_M_detach_all](#) ()
[protected, inherited]

Detach all iterators, leaving them singular.

5.242.2.2 void [__gnu_debug::Safe_sequence_base::_M_detach_singular](#) ()
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

5.242.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

5.242.2.4 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.242.2.5 `void __gnu_debug::Safe_sequence< deque< _Tp, _Allocator > >::M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.242.2.6 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.242.2.7 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.242.2.8 `void __gnu_debug::Safe_sequence< deque< _Tp, _Allocator > >::M_transfer_iter (const _Safe_iterator< _Iterator, deque< _Tp, _Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.242.3 Member Data Documentation

5.242.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.242.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.242.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

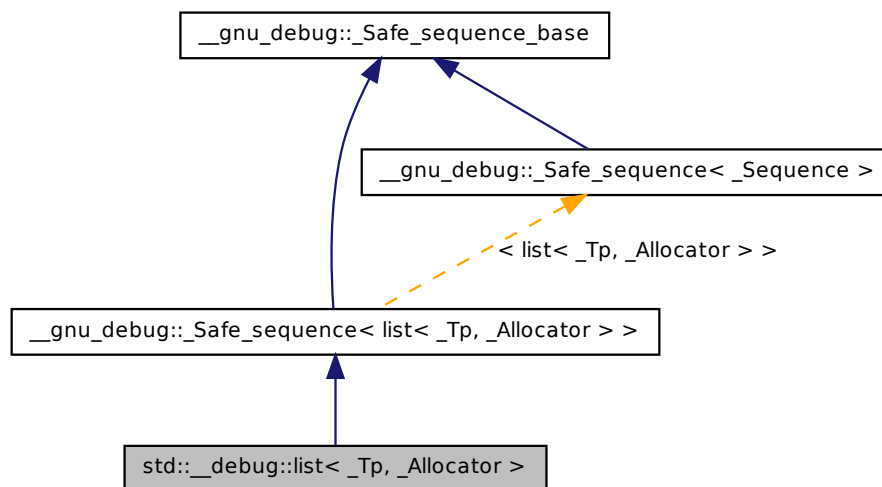
The documentation for this class was generated from the following file:

- [debug/deque](#)

5.243 `std::__debug::list< _Tp, _Allocator >` Class Template Reference

Class `std::list` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::list< _Tp, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, list >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **list** (const _Allocator &__a=_Allocator())
- **list** (size_type __n)
- template<class _InputIterator >
 list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **list** (const list &__x)
- **list** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- **list** (const _Base &__x)
- **list** (list &&__x)
- _Base & _M_base ()
- const _Base & _M_base () const
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_iter (const _Safe_iterator< _Iterator, list< _Tp, _Allocator > &__x)
- void **assign** (size_type __n, const _Tp &__t)
- void **assign** (initializer_list< value_type > __l)
- template<class _InputIterator >
 void **assign** (_InputIterator __first, _InputIterator __last)
- const_reference **back** () const
- reference **back** ()
- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... _Args>
 iterator **emplace** (iterator __position, _Args &&...__args)
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __position, iterator __last)
- const_reference **front** () const
- reference **front** ()
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, initializer_list< value_type > __l)

- `template<class _InputIterator >`
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `template<typename _Compare >`
`void merge (list &__x, _Compare __comp)`
- `void merge (list &&__x)`
- `void merge (list &__x)`
- `template<class _Compare >`
`void merge (list &&__x, _Compare __comp)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back ()`
- `void pop_front ()`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `void remove (const _Tp &__value)`
- `template<class _Predicate >`
`void remove_if (_Predicate __pred)`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const _Tp &__c)`
- `template<typename _StrictWeakOrdering >`
`void sort (_StrictWeakOrdering __pred)`
- `void sort ()`
- `void splice (iterator __position, list &&__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x, iterator __i)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void splice (iterator __position, list &__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &&__x)`
- `void splice (iterator __position, list &__x)`
- `void splice (iterator __position, list &__x)`
- `void swap (list &__x)`
- `template<class _BinaryPredicate >`
`void unique (_BinaryPredicate __binary_pred)`
- `void unique ()`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- __gnu_cxx::__mutex & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.243.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::list< _Tp, _Allocator >

Class [std::list](#) with safety/checking/debug instrumentation.

Definition at line 43 of file debug/list.

5.243.2 Member Function Documentation

5.243.2.1 void __gnu_debug::Safe_sequence_base::_M_detach_all ()
[protected, inherited]

Detach all iterators, leaving them singular.

5.243.2.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ()
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

5.243.2.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_
base::_M_get_mutex () throw () [protected,
inherited]

For use in [_Safe_sequence](#).

Referenced by [__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if\(\)](#),
and [__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter\(\)](#).

5.243.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.243.2.5 `void __gnu_debug::Safe_sequence< list< _Tp, _Allocator >
>::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.243.2.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.243.2.7 `void __gnu_debug::Safe_sequence_base::_M_swap (
_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.243.2.8 `void __gnu_debug::Safe_sequence< list< _Tp, _Allocator >
>::_M_transfer_iter (const _Safe_iterator< _Iterator, list< _Tp,
_Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.243.3 Member Data Documentation

5.243.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_
const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

5.244 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1339

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.243.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.243.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

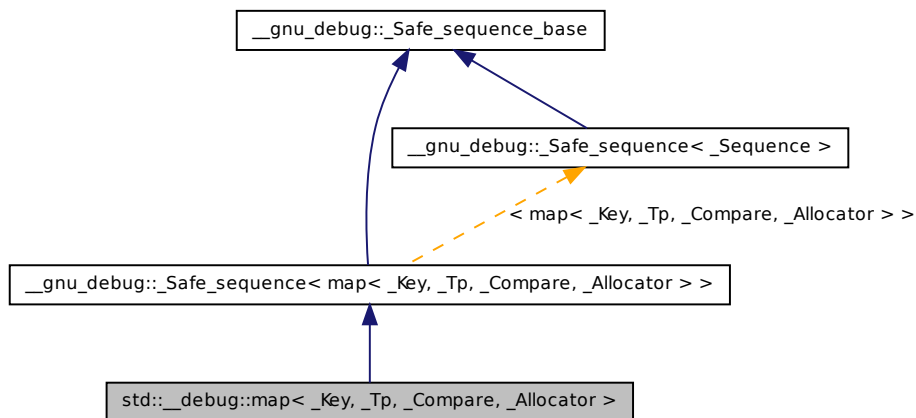
The documentation for this class was generated from the following file:

- [debug/list](#)

5.244 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::map` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, map >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, map >` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **map** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
map (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **map** (const `_Base` &__x)
- **map** (`map` &&__x)
- **map** (const `map` &__x)
- **map** (`initializer_list`< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_iter** (const `_Safe_iterator`< `_Iterator`, `map`< `_Key`, `_Tp`, `_Compare`, `_Allocator` > > &__x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x) const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)
- `const_iterator` **find** (const `key_type` &__x) const
- void **insert** (`std::initializer_list`< `value_type` > __list)
- `std::pair`< `iterator`, `bool` > **insert** (const `value_type` &__x)
- `iterator` **insert** (`iterator` __position, const `value_type` &__x)
- template<typename `_InputIterator` >
void **insert** (`_InputIterator` __first, `_InputIterator` __last)
- `iterator` **lower_bound** (const `key_type` &__x)
- `const_iterator` **lower_bound** (const `key_type` &__x) const
- `map` & **operator=** (const `map` &__x)

- `map` & `operator=` (`map` &&__x)
- `map` & `operator=` (`initializer_list`< `value_type` > __l)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- void `swap` (`map` &__x)
- `const_iterator` `upper_bound` (const key_type &__x) const
- `iterator` `upper_bound` (const key_type &__x)

Public Attributes

- `_Safe_iterator_base` * `_M_const_iterators`
- `_Safe_iterator_base` * `_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &__x)

5.244.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class `std::map` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/map.h`.

5.244.2 Member Function Documentation

5.244.2.1 void `__gnu_debug::_Safe_sequence_base::_M_detach_all` ()
[protected, inherited]

Detach all iterators, leaving them singular.

5.244 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference 1343

5.244.2.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular () [protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.244.2.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.244.2.4 void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.244.2.5 void __gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.244.2.6 void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.244.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (`
`_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.244.2.8 `void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare,`
`_Allocator > >::_M_transfer_iter (const _Safe_iterator<`
`_Iterator, map< _Key, _Tp, _Compare, _Allocator > > & __x)`
`[inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.244.3 Member Data Documentation

5.244.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_`
`const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.244.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_`
`iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.244.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version`
`[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

5.245 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1345

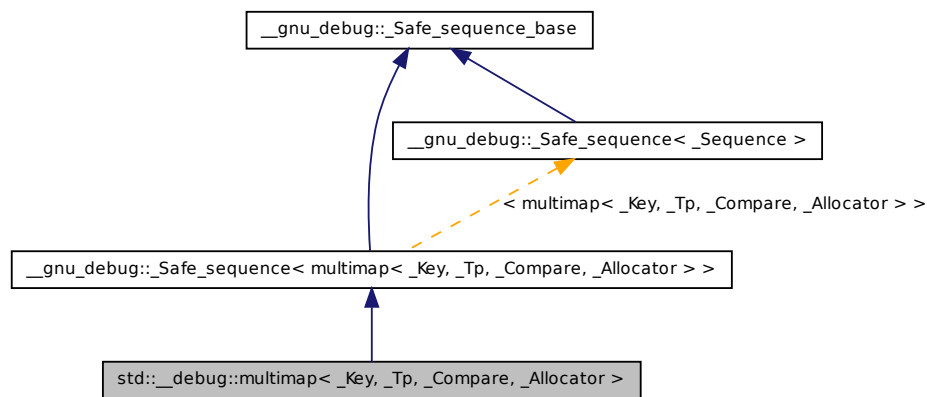
The documentation for this class was generated from the following file:

- [debug/map.h](#)

5.245 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::__Safe_iterator< typename _Base::const_iterator, multimap >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::__Safe_iterator< typename _Base::iterator, multimap >` **iterator**

- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **multimap** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
multimap (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **multimap** (const `_Base` &__x)
- **multimap** (`multimap` &&__x)
- **multimap** (const `multimap` &__x)
- **multimap** (`initializer_list`< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_iter** (const `_Safe_iterator`< `_Iterator`, `multimap`< `_Key`, `_Tp`, `_Compare`, `_Allocator` > > &__x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x) const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)

- `const_iterator find` (const key_type &__x) const
- void `insert` (std::initializer_list< value_type > __list)
- `iterator insert` (const value_type &__x)
- `iterator insert` (iterator __position, const value_type &__x)
- template<typename _InputIterator >
void `insert` (_InputIterator __first, _InputIterator __last)
- `iterator lower_bound` (const key_type &__x)
- `const_iterator lower_bound` (const key_type &__x) const
- `multimap & operator=` (const multimap &__x)
- `multimap & operator=` (multimap &&__x)
- `multimap & operator=` (initializer_list< value_type > __l)
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rend` () const
- `reverse_iterator rend` ()
- void `swap` (multimap &__x)
- `const_iterator upper_bound` (const key_type &__x) const
- `iterator upper_bound` (const key_type &__x)

Public Attributes

- _Safe_iterator_base * `_M_const_iterators`
- _Safe_iterator_base * `_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- __gnu_cxx::__mutex & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (_Safe_sequence_base &__x)

5.245.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class `std::multimap` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multimap.h`.

5.245.2 Member Function Documentation

5.245.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.245.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.245.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw ()` **[protected, inherited]**

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.245.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()`
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.245.2.5 `void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred)`
[inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.245.2.6 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.245.2.7 void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.245.2.8 void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::M_transfer_iter (const _Safe_iterator< _Iterator, multimap< _Key, _Tp, _Compare, _Allocator > & __x) [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.245.3 Member Data Documentation

5.245.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.245.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.245.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

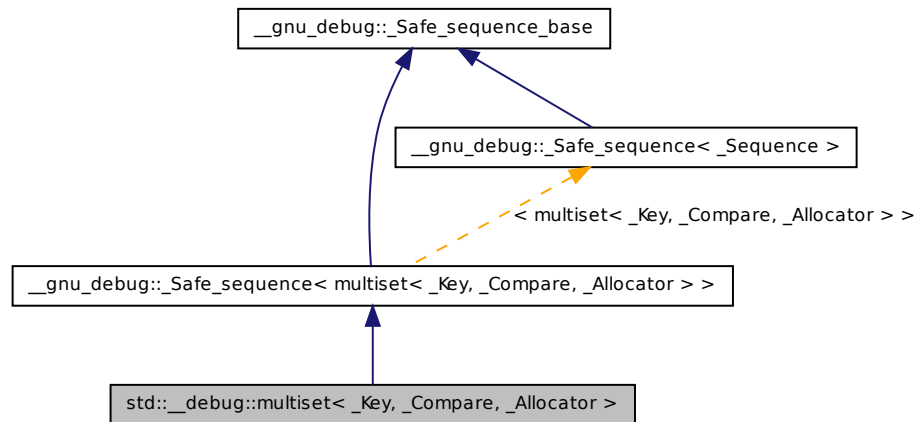
The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

5.246 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference

Class [std::multiset](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, multiset >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**

- typedef _Base::const_reference **const_reference**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef [__gnu_debug::_Safe_iterator](#)< typename _Base::iterator, [multiset](#) > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multiset** (const [_Base](#) &__x)
- **multiset** ([multiset](#) &&__x)
- **multiset** (const [multiset](#) &__x)
- **multiset** ([initializer_list](#)< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- [_Base](#) & **_M_base** ()
- const [_Base](#) & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (_Predicate __pred)
- void **_M_transfer_iter** (const [_Safe_iterator](#)< _Iterator, [multiset](#)< _Key, _Compare, _Allocator > > &__x)
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- [const_iterator](#) **cbegin** () const
- [const_iterator](#) **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal_range** (const key_type &__x)

- `std::pair< const_iterator, const_iterator > equal_range` (const key_type &__x) const
- `iterator erase` (iterator __first, iterator __last)
- `iterator erase` (iterator __position)
- `size_type erase` (const key_type &__x)
- `iterator find` (const key_type &__x)
- `const_iterator find` (const key_type &__x) const
- `iterator insert` (iterator __position, const value_type &__x)
- `iterator insert` (const value_type &__x)
- `template<typename _InputIterator >`
`void insert` (_InputIterator __first, _InputIterator __last)
- `void insert` (initializer_list< value_type > __l)
- `iterator lower_bound` (const key_type &__x)
- `const_iterator lower_bound` (const key_type &__x) const
- `multiset & operator=` (const multiset &__x)
- `multiset & operator=` (multiset &&__x)
- `multiset & operator=` (initializer_list< value_type > __l)
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rend` () const
- `reverse_iterator rend` ()
- `void swap` (multiset &__x)
- `const_iterator upper_bound` (const key_type &__x) const
- `iterator upper_bound` (const key_type &__x)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `void _M_revalidate_singular` ()
- `void _M_swap` (_Safe_sequence_base &__x)

5.246.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename
_Allocator = std::allocator<_Key>> class std::__debug::multiset< _Key, _-
_Compare, _Allocator >
```

Class [std::multiset](#) with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multiset.h`.

5.246.2 Member Function Documentation

5.246.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.246.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.246.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_-`
`base::_M_get_mutex () throw ()` [protected,
inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`,
and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.246.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()`
`const` [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.246.2.5 `void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred)`
[inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.246.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ()`
[protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.246.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x)`
[protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.246.2.8 `void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_transfer_iter (const _Safe_iterator< _Iterator, multiset< _Key, _Compare, _Allocator > > & __x)`
[inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.246.3 Member Data Documentation

5.246.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.247 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference 1355

5.246.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter().

5.246.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

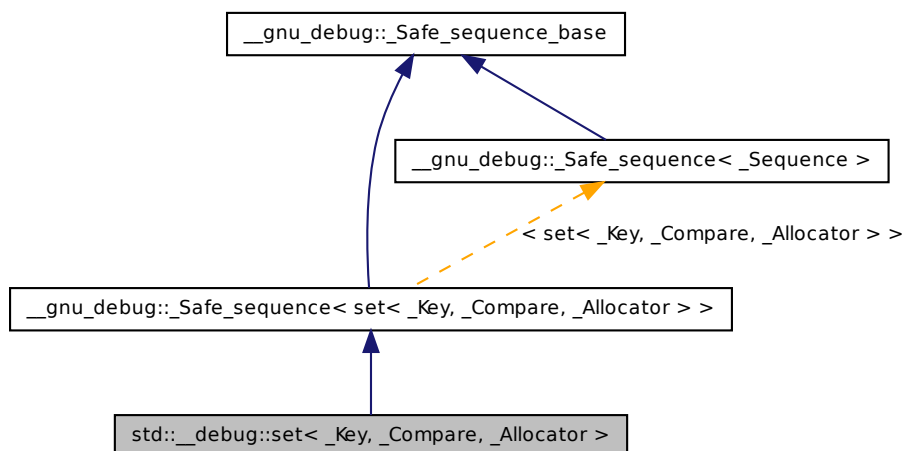
The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

5.247 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference

Class [std::set](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::set< _Key, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, set >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, set >` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **set** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- `template<typename _InputIterator >`
set (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **set** (const `_Base` &__x)
- **set** (`set` &&__x)
- **set** (const `set` &__x)
- **set** (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_iter** (const `_Safe_iterator`< `_Iterator`, `set`< `_Key`, `_Compare`, `_Allocator` > > &__x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x) const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)
- `const_iterator` **find** (const `key_type` &__x) const
- `iterator` **insert** (`iterator` __position, const `value_type` &__x)
- `std::pair`< `iterator`, `bool` > **insert** (const `value_type` &__x)
- `template<typename _InputIterator >`
void **insert** (`_InputIterator` __first, `_InputIterator` __last)
- void **insert** (`initializer_list`< `value_type` > __l)
- `iterator` **lower_bound** (const `key_type` &__x)
- `const_iterator` **lower_bound** (const `key_type` &__x) const
- `set` & **operator=** (const `set` &__x)

- `set & operator= (set &&__x)`
- `set & operator= (initializer_list< value_type > __l)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void swap (set &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `iterator upper_bound (const key_type &__x)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.247.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__debug::set< _Key, _Compare, _Allocator >`

Class `std::set` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/set.h`.

5.247.2 Member Function Documentation

5.247.2.1 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()` [protected, inherited]

Detach all iterators, leaving them singular.

5.247 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference 1359

5.247.2.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ()
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.247.2.3 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(), and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.247.2.4 void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe_base.h.

5.247.2.5 void __gnu_debug::Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred)
[inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred*(*x*) returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

5.247.2.6 void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.247.2.7 `void __gnu_debug::Safe_sequence_base::_M_swap (`
`_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.247.2.8 `void __gnu_debug::Safe_sequence< set< _Key, _Compare,`
`_Allocator > >::_M_transfer_iter (const _Safe_iterator< _Iterator,`
`set< _Key, _Compare, _Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.247.3 Member Data Documentation

5.247.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M-`
`const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.247.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M-`
`iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.247.3.3 `unsigned int __gnu_debug::Safe_sequence_base::_M_version`
`[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

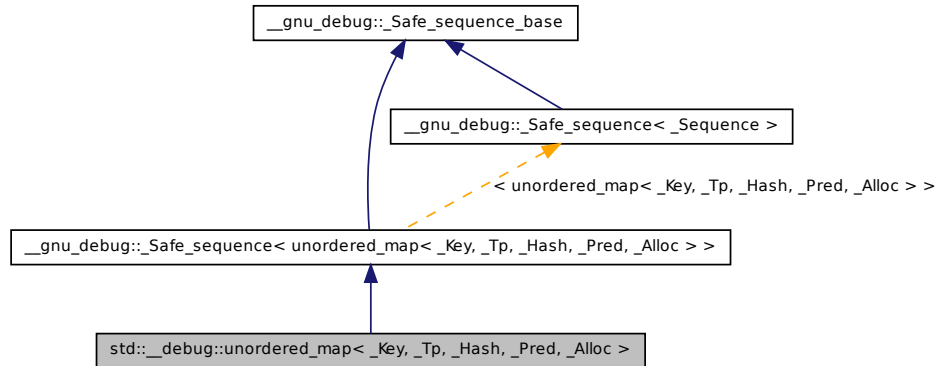
The documentation for this class was generated from the following file:

- [debug/set.h](#)

5.248 `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_map` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`:



Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_map > const_iterator`
- `typedef _Base::hasher hasher`
- `typedef __gnu_debug::Safe_iterator< typename _Base::iterator, unordered_map > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const [_Base](#) &__x)
- **unordered_map** ([unordered_map](#) &&__x)
- **unordered_map** (const [unordered_map](#) &__x)
- **unordered_map** ([initializer_list](#)< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- void [_M_invalidate_all](#) () const
- void [_M_invalidate_if](#) (_Predicate __pred)
- void [_M_transfer_iter](#) (const _Safe_iterator< _Iterator, [unordered_map](#)< _Key, _Tp, _Hash, _Pred, _Alloc > > &__x)
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- [const_iterator](#) **cbegin** () const
- [const_iterator](#) **cend** () const
- void **clear** ()
- [const_iterator](#) **end** () const
- [iterator](#) **end** ()
- [std::pair](#)< [iterator](#), [iterator](#) > **equal_range** (const key_type &__key)
- [std::pair](#)< [const_iterator](#), [const_iterator](#) > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- [iterator](#) **erase** ([const_iterator](#) __it)
- [iterator](#) **erase** ([const_iterator](#) __first, [const_iterator](#) __last)
- [iterator](#) **find** (const key_type &__key)
- [const_iterator](#) **find** (const key_type &__key) const
- [iterator](#) **insert** ([const_iterator](#), const value_type &__obj)
- [std::pair](#)< [iterator](#), bool > **insert** (const value_type &__obj)
- void **insert** ([std::initializer_list](#)< value_type > __l)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- [unordered_map](#) & **operator=** ([unordered_map](#) &&__x)
- [unordered_map](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_map](#) & **operator=** (const [unordered_map](#) &__x)
- void **swap** ([unordered_map](#) &__x)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.248.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_
Key>> class std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>
```

Class `std::unordered_map` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_map`.

5.248.2 Member Function Documentation

5.248.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.248.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.248.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.248.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.248.2.5 `void __gnu_debug::_Safe_sequence<unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>>::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.248.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.248.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.248 std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> Class Template Reference 1365

5.248.2.8 void __gnu_debug::_Safe_sequence< unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> >::_M_transfer_iter (const _Safe_iterator<_Iterator, unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> > & __x) **[inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.248.3 Member Data Documentation

5.248.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if(), __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single(), and __gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter().

5.248.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators **[inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if(), __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single(), and __gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter().

5.248.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version **[mutable, inherited]**

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

The documentation for this class was generated from the following file:

- [debug/unordered_map](#)

5.249 `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:



Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_multimap >` **const_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_multimap >` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_multimap** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename InputIterator >`
unordered_multimap (`InputIterator __f`, `InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_multimap** (`const _Base &__x`)
- **unordered_multimap** (`unordered_multimap &&__x`)
- **unordered_multimap** (`const unordered_multimap &__x`)
- **unordered_multimap** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`

- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> > &__x)`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `const_iterator end () const`
- `iterator end ()`
- `std::pair< iterator, iterator > equal_range (const key_type &__key)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `size_type erase (const key_type &__key)`
- `iterator erase (const_iterator __it)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__key)`
- `const_iterator find (const key_type &__key) const`
- `iterator insert (const_iterator, const value_type &__obj)`
- `iterator insert (const value_type &__obj)`
- `void insert (std::initializer_list< value_type > __l)`
- `template<typename _InputIterator >
void insert (_InputIterator __first, _InputIterator __last)`
- `unordered_multimap & operator= (unordered_multimap &&__x)`
- `unordered_multimap & operator= (initializer_list< value_type > __l)`
- `unordered_multimap & operator= (const unordered_multimap &__x)`
- `void swap (unordered_multimap &__x)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.249.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Definition at line 308 of file `debug/unordered_map`.

5.249.2 Member Function Documentation

5.249.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.249.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.249.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw ()` [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.249.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()`
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.249.2.5 void __gnu_debug::Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_invalidate_if (_Predicate __pred) **[inherited]**

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred(x)* returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

5.249.2.6 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () **[protected, inherited]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.249.2.7 void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) **[protected, inherited]**

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.249.2.8 void __gnu_debug::Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) **[inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.249.3 Member Data Documentation

5.249.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

5.249.3.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.249.3.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` `[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/unordered_map](#)

5.250 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`:



Public Types

- `typedef _Base::allocator_type` **allocator_type**
- `typedef __gnu_debug::_Safe_iterator< typename _Base::const_iterator, unordered_multiset >` **const_iterator**
- `typedef _Base::hasher` **hasher**
- `typedef __gnu_debug::_Safe_iterator< typename _Base::iterator, unordered_multiset >` **iterator**
- `typedef _Base::key_equal` **key_equal**
- `typedef _Base::key_type` **key_type**

- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multiset (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const _Base &__x)
- **unordered_multiset** (unordered_multiset &&__x)
- **unordered_multiset** (const unordered_multiset &__x)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** ()
- const _Base & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (_Predicate __pred)
- void **_M_transfer_iter** (const _Safe_iterator< _Iterator, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x)
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_iterator **end** () const
- iterator **end** ()
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __it)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- iterator **insert** (const_iterator, const value_type &__obj)
- iterator **insert** (const value_type &__obj)
- void **insert** (std::initializer_list< value_type > __l)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)

- `unordered_multiset` & `operator=` (`unordered_multiset` &&__x)
- `unordered_multiset` & `operator=` (`initializer_list`< `value_type` > __l)
- `unordered_multiset` & `operator=` (const `unordered_multiset` &__x)
- void `swap` (`unordered_multiset` &__x)

Public Attributes

- `_Safe_iterator_base` * `_M_const_iterators`
- `_Safe_iterator_base` * `_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &__x)

5.250.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 305 of file `debug/unordered_set`.

5.250.2 Member Function Documentation

5.250.2.1 void `__gnu_debug::_Safe_sequence_base::_M_detach_all` ()
[protected, inherited]

Detach all iterators, leaving them singular.

5.250.2.2 void `__gnu_debug::_Safe_sequence_base::_M_detach_singular` ()
[protected, inherited]

Detach all singular iterators.

5.250 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference 1373

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.250.2.3 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`,
and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.250.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.250.2.5 `void __gnu_debug::Safe_sequence< unordered_multiset< _Value,
_Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred)
[inherited]`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.250.2.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.250.2.7 `void __gnu_debug::Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.250.2.8 `void __gnu_debug::_Safe_sequence< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multiset< _Value, _Hash, _Pred, _Alloc > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.250.3 Member Data Documentation

5.250.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.250.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.250.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

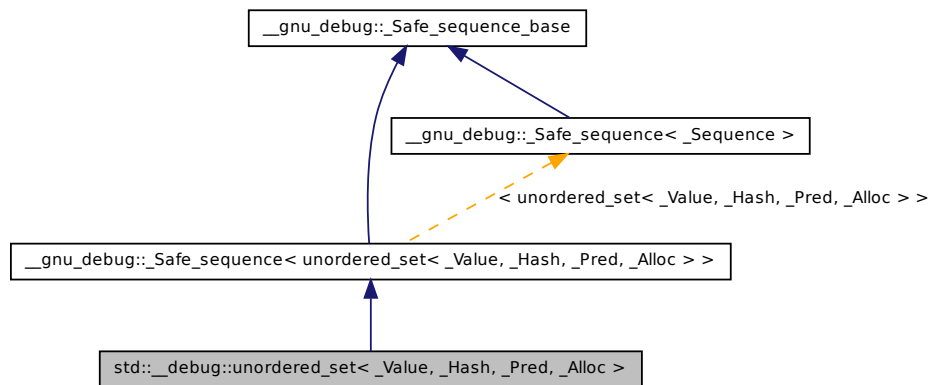
The documentation for this class was generated from the following file:

- [debug/unordered_set](#)

5.251 `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_set` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`:



Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_set >` **const_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_set >` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_set** (`size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)

- `template<typename _InputIterator >`
`unordered_set` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_set` (`const _Base &__x`)
- `unordered_set` (`unordered_set &&__x`)
- `unordered_set` (`const unordered_set &__x`)
- `unordered_set` (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, unordered_set< _Value, _Hash, _Pred, _Alloc > > &__x)`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `const_iterator end () const`
- `iterator end ()`
- `std::pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__key`) `const`
- `size_type erase` (`const key_type &__key`)
- `iterator erase` (`const_iterator __it`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const_iterator`, `const value_type &__obj`)
- `std::pair< iterator, bool > insert` (`const value_type &__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _InputIterator >`
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `unordered_set & operator=` (`unordered_set &&__x`)
- `unordered_set & operator=` (`initializer_list< value_type > __l`)
- `unordered_set & operator=` (`const unordered_set &__x`)
- `void swap` (`unordered_set &__x`)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

5.251.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>
```

Class `std::unordered_set` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_set`.

5.251.2 Member Function Documentation

5.251.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.251.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.251.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.251.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.251.2.5 `void __gnu_debug::_Safe_sequence< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.251.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.251.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.251.2.8 void __gnu_debug::_Safe_sequence< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_transfer_iter (const _Safe_iterator< _Iterator, unordered_set< _Value, _Hash, _Pred, _Alloc > > & __x)
 [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.251.3 Member Data Documentation

5.251.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter().

5.251.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter().

5.251.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

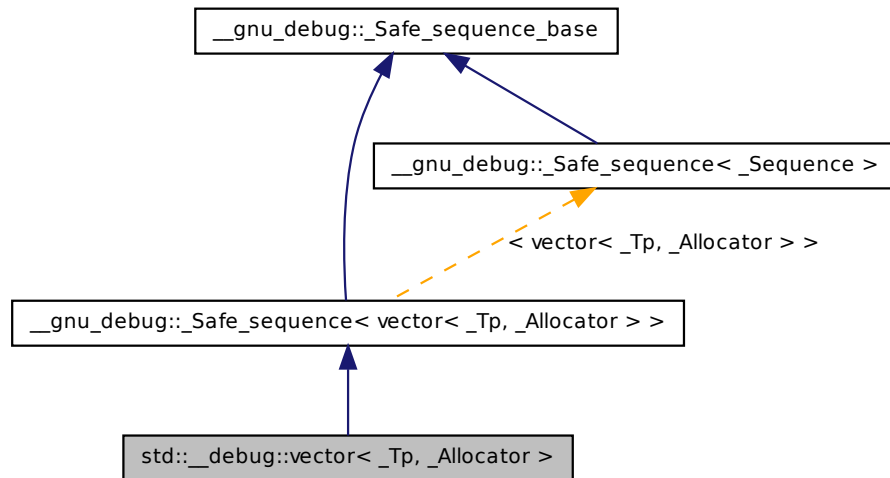
The documentation for this class was generated from the following file:

- [debug/unordered_set](#)

5.252 `std::__debug::vector< _Tp, _Allocator >` Class Template Reference

Class `std::vector` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::vector< _Tp, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, vector >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**

- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **vector** (const _Allocator &__a=_Allocator())
- **vector** (size_type __n)
- template<class _InputIterator >
 vector (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **vector** ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- **vector** (const [vector](#) &__x)
- **vector** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- [vector](#) (const _Base &__x)
- **vector** ([vector](#) &&__x)
- _Base & _M_base ()
- const _Base & _M_base () const
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_iter (const _Safe_iterator< _Iterator, [vector](#)< _Tp, _Allocator > > &__x)
- template<typename _InputIterator >
 void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__u)
- void **assign** ([initializer_list](#)< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- [iterator](#) **begin** ()
- const_iterator **begin** () const
- size_type **capacity** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... _Args>
 [iterator](#) **emplace** ([iterator](#) __position, _Args &&...__args)
- template<typename... _Args>
 void **emplace_back** (_Args &&...__args)
- [iterator](#) **end** ()
- const_iterator **end** () const

- [iterator erase](#) ([iterator](#) __first, [iterator](#) __last)
- [iterator erase](#) ([iterator](#) __position)
- reference **front** ()
- const_reference **front** () const
- void **insert** ([iterator](#) __position, [initializer_list](#)< value_type > __l)
- void **insert** ([iterator](#) __position, size_type __n, const _Tp &__x)
- [iterator insert](#) ([iterator](#) __position, const _Tp &__x)
- template<typename _Up = _Tp>
__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, [iterator](#) >::__type **insert** ([iterator](#) __position, _Tp &&__x)
- template<class _InputIterator >
void **insert** ([iterator](#) __position, _InputIterator __first, _InputIterator __last)
- [vector](#) & **operator=** (const [vector](#) &__x)
- [vector](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [vector](#) & **operator=** ([vector](#) &&__x)
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- void **pop_back** ()
- void **push_back** (const _Tp &__x)
- template<typename _Up = _Tp>
__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, void >::__type **push_back** (_Tp &&__x)
- [reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- void **reserve** (size_type __n)
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **swap** ([vector](#) &__x)

Public Attributes

- _Safe_iterator_base * [_M_const_iterators](#)
- _Safe_iterator_base * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- __gnu_cxx::__mutex & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.252.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::vector< _Tp, _Allocator >
```

Class [std::vector](#) with safety/checking/debug instrumentation.

Definition at line 45 of file debug/vector.

5.252.2 Constructor & Destructor Documentation

```
5.252.2.1 template<typename _Tp, typename _Allocator =
std::allocator<_Tp>> std::__debug::vector< _Tp, _Allocator
>::vector ( const_Base & __x ) [inline]
```

Construction from a release-mode vector.

Definition at line 106 of file debug/vector.

5.252.3 Member Function Documentation

```
5.252.3.1 void __gnu_debug::Safe_sequence_base::_M_detach_all ( )
[protected, inherited]
```

Detach all iterators, leaving them singular.

```
5.252.3.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )
[protected, inherited]
```

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

```
5.252.3.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_-
base::_M_get_mutex ( ) throw () [protected,
inherited]
```

For use in [_Safe_sequence](#).

Referenced by [__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if\(\)](#),
and [__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter\(\)](#).

5.252.3.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.252.3.5 `void __gnu_debug::Safe_sequence< vector< _Tp, _Allocator >
>::_M_invalidate_if(_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.252.3.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.252.3.7 `void __gnu_debug::Safe_sequence_base::_M_swap (
_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.252.3.8 `void __gnu_debug::Safe_sequence< vector< _Tp, _Allocator >
>::_M_transfer_iter(const _Safe_iterator< _Iterator, vector< _Tp,
_Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.252.4 Member Data Documentation

5.252.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_
const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.252.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.252.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

5.253 `std::__declval_protector< _Tp >` Struct Template Reference

`declval`

Static Public Member Functions

- static [add_rvalue_reference< _Tp >::type __delegate \(\)](#)

Static Public Attributes

- static const bool `__stop`

5.253.1 Detailed Description

`template<typename _Tp> struct std::__declval_protector< _Tp >`

declval

Definition at line 672 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.254 `std::__exception_ptr::exception_ptr` Class Reference

An opaque pointer to an arbitrary exception.

Public Member Functions

- `exception_ptr` (const [exception_ptr](#) &) throw ()
- `exception_ptr` ([exception_ptr](#) &&__o) throw ()
- `exception_ptr` (nullptr_t) throw ()
- const [type_info](#) * `__cxa_exception_type` () const __attribute__((__pure__)) throw ()
- `operator bool` () const
- [exception_ptr](#) & `operator=` (const [exception_ptr](#) &) throw ()
- [exception_ptr](#) & `operator=` ([exception_ptr](#) &&__o) throw ()
- void `swap` ([exception_ptr](#) &) throw ()

Friends

- bool `operator==` (const [exception_ptr](#) &, const [exception_ptr](#) &) __attribute__((__pure__)) throw ()
- [exception_ptr](#) `std::current_exception` () throw ()
- void `std::rethrow_exception` ([exception_ptr](#))

5.254.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 73 of file `exception_ptr.h`.

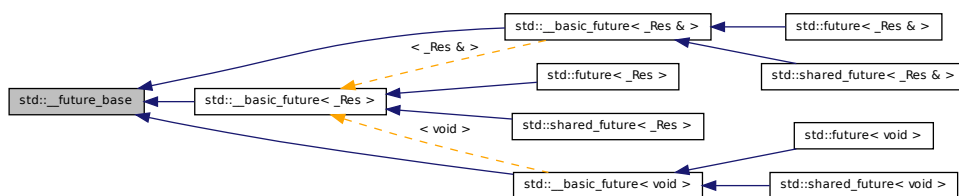
The documentation for this class was generated from the following file:

- [exception_ptr.h](#)

5.255 std::__future_base Struct Reference

Base class and enclosing scope.

Inheritance diagram for std::__future_base:



Classes

- [struct _Ptr](#)
A [unique_ptr](#) based on the instantiating type.
- [struct _Result](#)
Result.
- [struct _Result<_Res &>](#)
Partial specialization for reference types.
- [struct _Result<void>](#)
Explicit specialization for void.
- [struct _Result_base](#)
Base class for results.
- [class _State](#)
Shared state between a promise and one or more associated futures.

5.255.1 Detailed Description

Base class and enclosing scope.

Definition at line 136 of file future.

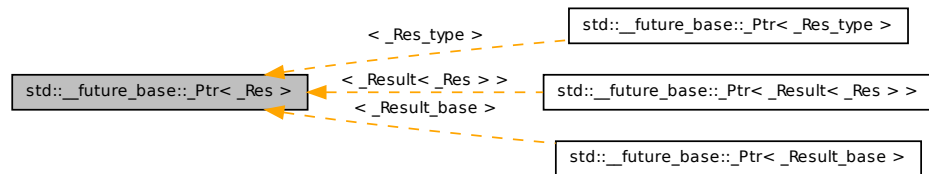
The documentation for this struct was generated from the following file:

- [future](#)

5.256 std::__future_base::_Ptr< _Res > Struct Template Reference

A [unique_ptr](#) based on the instantiating type.

Inheritance diagram for std::__future_base::_Ptr< _Res >:



Public Types

- typedef [unique_ptr](#)< _Res, _Result_base::_Deleter > **type**

5.256.1 Detailed Description

template<typename _Res> struct std::__future_base::_Ptr< _Res >

A [unique_ptr](#) based on the instantiating type.

Definition at line 211 of file future.

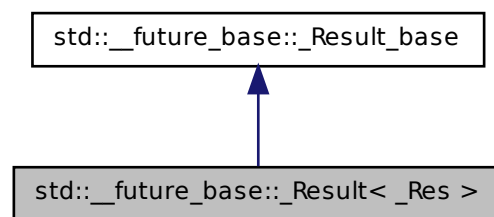
The documentation for this struct was generated from the following file:

- [future](#)

5.257 `std::__future_base::_Result< _Res >` Struct Template Reference

Result.

Inheritance diagram for `std::__future_base::_Result< _Res >`:



Public Member Functions

- `void _M_set (const _Res &__res)`
- `void _M_set (_Res &&__res)`
- `_Res & _M_value ()`

Public Attributes

- `exception_ptr _M_error`

5.257.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Result< _Res >`

Result.

Definition at line 161 of file `future`.

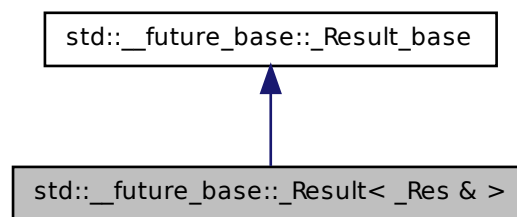
The documentation for this struct was generated from the following file:

- [future](#)

5.258 `std::__future_base::_Result< _Res & >` Struct Template Reference

Partial specialization for reference types.

Inheritance diagram for `std::__future_base::_Result< _Res & >`:



Public Member Functions

- `_Res & _M_get ()`
- `void _M_set (_Res & __res)`

Public Attributes

- `exception_ptr _M_error`

5.258.1 Detailed Description

template<typename _Res> struct `std::__future_base::_Result< _Res & >`

Partial specialization for reference types.

Definition at line 455 of file `future`.

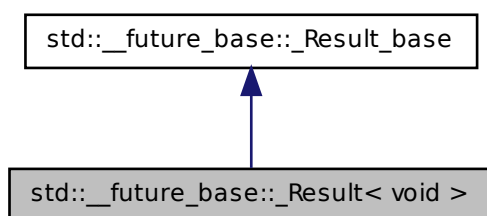
The documentation for this struct was generated from the following file:

- [future](#)

5.259 `std::__future_base::_Result< void >` Struct Template Reference

Explicit specialization for void.

Inheritance diagram for `std::__future_base::_Result< void >`:



Public Attributes

- `exception_ptr _M_error`

5.259.1 Detailed Description

`template<> struct std::__future_base::_Result< void >`

Explicit specialization for void.

Definition at line 471 of file `future`.

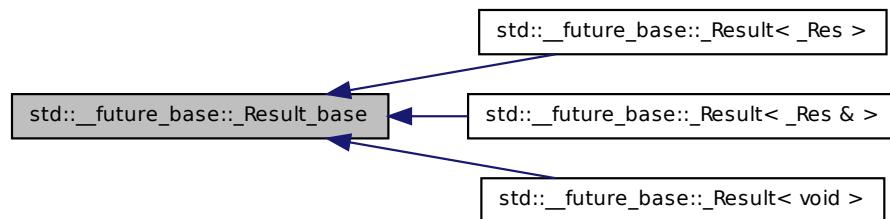
The documentation for this struct was generated from the following file:

- [future](#)

5.260 `std::__future_base::_Result_base` Struct Reference

Base class for results.

Inheritance diagram for `std::__future_base::_Result_base`:



Public Member Functions

- `_Result_base` (const [_Result_base](#) &)
- virtual void `_M_destroy` ()=0
- [_Result_base](#) & `operator=` (const [_Result_base](#) &)

Public Attributes

- `exception_ptr _M_error`

5.260.1 Detailed Description

Base class for results.

Definition at line 139 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

5.261 std::__future_base::_State Class Reference

Shared state between a promise and one or more associated futures.

Inherited by `std::__future_base::_Async_state<_Res>`, `std::__future_base::_Deferred_state<_Res>`, and `std::__future_base::_Task_state<_Res(_Args...)>`.

Public Member Functions

- [_State](#) (const [_State](#) &)
- void [_M_break_promise](#) (_Ptr_type __res)
- void [_M_set_result](#) (function< _Ptr_type()> __res, bool __ignore_failure=false)
- void [_M_set_retrieved_flag](#) ()
- [_State](#) & [operator=](#) (const [_State](#) &)
- [_Result_base](#) & [wait](#) ()
- template<typename _Rep, typename _Period >
bool [wait_for](#) (const [chrono::duration](#)< _Rep, _Period > &__rel)
- template<typename _Clock, typename _Duration >
bool [wait_until](#) (const [chrono::time_point](#)< _Clock, _Duration > &__abs)

Static Public Member Functions

- template<typename _Res, typename _Arg >
static _Setter< _Res, _Arg && > [__setter](#) ([promise](#)< _Res > *__prom, _Arg &&__arg)
- template<typename _Res >
static _Setter< _Res, __exception_ptr_tag > [__setter](#) (exception_ptr &__ex, [promise](#)< _Res > *__prom)
- static _Setter< void, void > [__setter](#) ([promise](#)< void > *__prom)
- template<typename _Tp >
static bool [_S_check](#) (const [shared_ptr](#)< _Tp > &__p)

5.261.1 Detailed Description

Shared state between a promise and one or more associated futures.

Definition at line 262 of file future.

The documentation for this class was generated from the following file:

- [future](#)

5.262 std::__is_location_invariant< _Tp > Struct Template Reference

Inherits [integral_constant](#)< bool,(is_pointer< _Tp >::value||is_member_pointer< _Tp >::value)>.

Inherited by [std::__is_location_invariant< _Simple_type_wrapper< _Tp > >](#).

5.262.1 Detailed Description

template<typename _Tp> struct std::__is_location_invariant< _Tp >

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Also implies a trivial copy constructor and assignment operator.

Definition at line 1392 of file functional.

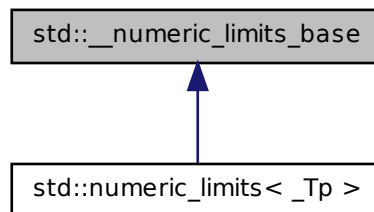
The documentation for this struct was generated from the following file:

- [functional](#)

5.263 std::__numeric_limits_base Struct Reference

Part of [std::numeric_limits](#).

Inheritance diagram for std::__numeric_limits_base:



Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const [float_denorm_style](#) [has_denorm](#)
- static const bool [has_denorm_loss](#)
- static const bool [has_infinity](#)
- static const bool [has_quiet_NaN](#)
- static const bool [has_signaling_NaN](#)

- static const bool `is_bounded`
- static const bool `is_exact`
- static const bool `is_iec559`
- static const bool `is_integer`
- static const bool `is_modulo`
- static const bool `is_signed`
- static const bool `is_specialized`
- static const int `max_digits10`
- static const int `max_exponent`
- static const int `max_exponent10`
- static const int `min_exponent`
- static const int `min_exponent10`
- static const int `radix`
- static const `float_round_style` `round_style`
- static const bool `tinyness_before`
- static const bool `traps`

5.263.1 Detailed Description

Part of `std::numeric_limits`. The `static const` members are usable as integral constant expressions.

Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the `std::numeric_limits` class.

Definition at line 190 of file `limits`.

5.263.2 Member Data Documentation

5.263.2.1 `const int std::__numeric_limits_base::digits` `[static]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 199 of file `limits`.

5.263.2.2 `const int std::__numeric_limits_base::digits10` `[static]`

The number of base 10 digits that can be represented without change.

Definition at line 201 of file `limits`.

5.263.2.3 `const float_denorm_style std::__numeric_limits_base::has_denorm [static]`

See [std::float_denorm_style](#) for more information.

Definition at line 244 of file limits.

5.263.2.4 `const bool std::__numeric_limits_base::has_denorm_loss [static]`

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42

Definition at line 247 of file limits.

5.263.2.5 `const bool std::__numeric_limits_base::has_infinity [static]`

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

5.263.2.6 `const bool std::__numeric_limits_base::has_quiet_NaN [static]`

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 239 of file limits.

5.263.2.7 `const bool std::__numeric_limits_base::has_signaling_NaN [static]`

True if the type has a representation for a signaling *Not a Number*.

Definition at line 242 of file limits.

5.263.2.8 `const bool std::__numeric_limits_base::is_bounded [static]`

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54

Definition at line 255 of file limits.

5.263.2.9 `const bool std::__numeric_limits_base::is_exact [static]`

True if the type uses an exact representation. *All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are*

exact but not integer. [18.2.1.2]/15

Definition at line 216 of file limits.

5.263.2.10 const bool std::__numeric_limits_base::is_iec559 [static]

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 251 of file limits.

5.263.2.11 const bool std::__numeric_limits_base::is_integer [static]

True if the type is integer. Is this supposed to be *if the type is integral*?

Definition at line 211 of file limits.

5.263.2.12 const bool std::__numeric_limits_base::is_modulo [static]

True if the type is *modulo*, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 260 of file limits.

5.263.2.13 const bool std::__numeric_limits_base::is_signed [static]

True if the type is signed.

Definition at line 208 of file limits.

5.263.2.14 const bool std::__numeric_limits_base::is_specialized [static]

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 194 of file limits.

5.263.2.15 const int std::__numeric_limits_base::max_digits10 [static]

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 205 of file limits.

5.263.2.16 `const int std::__numeric_limits_base::max_exponent` `[static]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 230 of file `limits`.

5.263.2.17 `const int std::__numeric_limits_base::max_exponent10` `[static]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 233 of file `limits`.

5.263.2.18 `const int std::__numeric_limits_base::min_exponent` `[static]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 223 of file `limits`.

5.263.2.19 `const int std::__numeric_limits_base::min_exponent10` `[static]`

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 226 of file `limits`.

5.263.2.20 `const int std::__numeric_limits_base::radix` `[static]`

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 219 of file `limits`.

5.263.2.21 `const float_round_style std::__numeric_limits_base::round_style`
`[static]`

See [std::float_round_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 269 of file `limits`.

5.264 `std::__parallel::_CRandNumber< _MustBeInt >` Struct Template Reference 1399

5.263.2.22 `const bool std::__numeric_limits_base::tinyness_before` `[static]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file `limits`.

5.263.2.23 `const bool std::__numeric_limits_base::traps` `[static]`

True if trapping is implemented for this type.

Definition at line 263 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.264 `std::__parallel::_CRandNumber< _MustBeInt >` Struct Template Reference

Functor wrapper for `std::rand()`.

Public Member Functions

- `int operator() (int __limit)`

5.264.1 Detailed Description

```
template<typename _MustBeInt = int> struct std::__parallel::_  
CRandNumber< _MustBeInt >
```

Functor wrapper for `std::rand()`.

Definition at line 1656 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

5.265 `std::__profile::bitset< _Nb >` Class Template Reference

Class [std::bitset](#) wrapper with performance instrumentation.

Inherits `bitset<_Nb>`.

Public Member Functions

- **bitset** (unsigned long long __val)
- `template<class _CharT, class _Traits, class _Alloc >`
bitset (const `std::basic_string<_CharT, _Traits, _Alloc>` &__str, typename `std::basic_string<_CharT, _Traits, _Alloc>::size_type` __pos, typename `std::basic_string<_CharT, _Traits, _Alloc>::size_type` __n, `_CharT` __zero, `_CharT` __one=`_CharT('1')`)
- **bitset** (const `_Base` &__x)
- `template<typename _CharT, typename _Traits, typename _Alloc >`
bitset (const `std::basic_string<_CharT, _Traits, _Alloc>` &__str, typename `std::basic_string<_CharT, _Traits, _Alloc>::size_type` __pos=0, typename `std::basic_string<_CharT, _Traits, _Alloc>::size_type` __n=(`std::basic_string<_CharT, _Traits, _Alloc>::npos`))
- **bitset** (const char *__str)
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- `bitset<_Nb>` & **flip** ()
- `bitset<_Nb>` & **flip** (size_t __pos)
- bool **operator!=** (const `bitset<_Nb>` &__rhs) const
- `bitset<_Nb>` & **operator&=** (const `bitset<_Nb>` &__rhs)
- `bitset<_Nb>` & **operator<<** (size_t __pos) const
- `bitset<_Nb>` & **operator<<=** (size_t __pos)
- bool **operator==** (const `bitset<_Nb>` &__rhs) const
- `bitset<_Nb>` & **operator>>** (size_t __pos) const
- `bitset<_Nb>` & **operator>>=** (size_t __pos)
- reference **operator[]** (size_t __pos)
- bool **operator[]** (size_t __pos) const
- `bitset<_Nb>` & **operator^=** (const `bitset<_Nb>` &__rhs)
- `bitset<_Nb>` & **operator|=** (const `bitset<_Nb>` &__rhs)
- `bitset<_Nb>` & **operator~** () const
- `bitset<_Nb>` & **reset** ()
- `bitset<_Nb>` & **reset** (size_t __pos)
- `bitset<_Nb>` & **set** (size_t __pos, bool __val=true)
- `bitset<_Nb>` & **set** ()
- `std::basic_string<char, std::char_traits<char>, std::allocator<char>>` **to_string** (char __zero, char __one= '1') const
- `template<typename _CharT, typename _Traits >`
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>>` **to_string** () const

5.266 `std::__profile::deque<_Tp, _Allocator>` Class Template Reference 1401

- `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc> to_string () const`
- `template<class _CharT, class _Traits>
std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string (_-
CharT __zero, _CharT __one=_CharT('1')) const`
- `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_-
string () const`
- `template<typename _CharT>
std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>
>> to_string () const`
- `template<class _CharT>
std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>
>> to_string (_CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT, class _Traits, class _Alloc>
std::basic_string<_CharT, _Traits, _Alloc> to_string (_CharT __zero, _CharT
__one=_CharT('1')) const`

5.265.1 Detailed Description

`template<size_t _Nb> class std::__profile::bitset<_Nb>`

Class `std::bitset` wrapper with performance instrumentation.

Definition at line 40 of file `profile/bitset`.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

5.266 `std::__profile::deque<_Tp, _Allocator>` Class Template Reference

Class `std::deque` wrapper with performance instrumentation.

Inherits `deque<_Tp, _Allocator>`.

Public Types

- `typedef _Allocator allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_pointer const_pointer`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::const_reverse_iterator const_reverse_iterator`

- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **deque** (const `_Allocator` &__a=_Allocator())
- **deque** (size_type __n)
- template<class `_InputIterator` >
deque (`_InputIterator` __first, `_InputIterator` __last, const `_Allocator` &__a=_Allocator())
- **deque** ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- **deque** (const [deque](#) &__x)
- **deque** (size_type __n, const `_Tp` &__value, const `_Allocator` &__a=_Allocator())
- **deque** (const [_Base](#) &__x)
- **deque** ([deque](#) &&__x)
- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- template<class `_InputIterator` >
void **assign** (`_InputIterator` __first, `_InputIterator` __last)
- void **assign** (size_type __n, const `_Tp` &__t)
- void **assign** ([initializer_list](#)< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... `_Args`>
iterator **emplace** (iterator __position, `_Args` &&...__args)
- template<typename... `_Args`>
void **emplace_back** (`_Args` &&...__args)
- template<typename... `_Args`>
void **emplace_front** (`_Args` &&...__args)

- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, [initializer_list](#)< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- iterator **insert** (iterator __position, const _Tp &__x)
- template<class _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- [deque](#) & **operator=** (const [deque](#) &__x)
- [deque](#) & **operator=** ([deque](#) &&__x)
- [deque](#) & **operator=** ([initializer_list](#)< value_type > __l)
- const_reference **operator[]** (size_type __n) const
- reference **operator[]** (size_type __n)
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (const _Tp &__x)
- void **push_back** (_Tp &&__x)
- void **push_front** (_Tp &&__x)
- void **push_front** (const _Tp &__x)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- const_reverse_iterator **rend** () const
- reverse_iterator **rend** ()
- void **resize** (size_type __sz, const _Tp &__c)
- void **resize** (size_type __sz)
- void **swap** ([deque](#) &__x)

5.266.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__profile::deque<_Tp, _Allocator>`

Class `std::deque` wrapper with performance instrumentation.

Definition at line 40 of file `profile/deque`.

The documentation for this class was generated from the following file:

- [profile/deque](#)

5.267 `std::__profile::list< _Tp, _Allocator >` Class Template Reference

List wrapper with performance instrumentation.

Inherits `list< _Tp, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__iterator_tracker< typename _Base::const_iterator, list >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__iterator_tracker< typename _Base::iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **list** (const `_Allocator` &__a=_Allocator())
- **list** (size_type __n)
- template<class `_InputIterator` >
 list (`_InputIterator` __first, `_InputIterator` __last, const `_Allocator` &__a=_Allocator())
- **list** ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- **list** (const [list](#) &__x)
- **list** (size_type __n, const `_Tp` &__value, const `_Allocator` &__a=_Allocator())
- **list** (const [_Base](#) &__x)
- **list** ([list](#) &&__x)
- const [_Base](#) & **_M_base** () const
- [_Base](#) & **_M_base** ()
- void **_M_profile_find** () const
- void **_M_profile_iterate** (int __rewind=0) const
- void **assign** ([initializer_list](#)< value_type > __l)

- template<class _InputIterator >
void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- const_reference **back** () const
- reference **back** ()
- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... _Args>
iterator **emplace** (iterator __position, _Args &&... __args)
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __position, iterator __last)
- const_reference **front** () const
- reference **front** ()
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- void **insert** (iterator __position, initializer_list< value_type > __l)
- template<class _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- template<class _Compare >
void **merge** (list &&__x, _Compare __comp)
- template<typename _Compare >
void **merge** (list &__x, _Compare __comp)
- void **merge** (list &&__x)
- void **merge** (list &__x)
- list & **operator=** (const list &__x)
- list & **operator=** (list &&__x)
- list & **operator=** (initializer_list< value_type > __l)
- void **pop_back** ()
- void **pop_front** ()
- void **push_front** (const value_type &__x)
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rbegin** ()
- void **remove** (const _Tp &__value)
- template<class _Predicate >
void **remove_if** (_Predicate __pred)

- [const_reverse_iterator](#) **rend** () const
- [reverse_iterator](#) **rend** ()
- void **resize** (size_type __sz, const _Tp &__c)
- void **resize** (size_type __sz)
- template<typename _StrictWeakOrdering >
void **sort** (_StrictWeakOrdering __pred)
- void **sort** ()
- void **splice** (iterator __position, [list](#) &&__x)
- void **splice** (iterator __position, [list](#) &&__x, iterator __first, iterator __last)
- void **splice** (iterator __position, [list](#) &__x, iterator __i)
- void **splice** (iterator __position, [list](#) &&__x, iterator __i)
- void **splice** (iterator __position, [list](#) &__x, iterator __first, iterator __last)
- void **splice** (iterator __position, [list](#) &__x)
- void **swap** ([list](#) &__x)
- template<class _BinaryPredicate >
void **unique** (_BinaryPredicate __binary_pred)
- void **unique** ()

5.267.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__profile::list< _Tp, _Allocator >

List wrapper with performance instrumentation.

Definition at line 42 of file profile/list.

The documentation for this class was generated from the following file:

- [profile/list](#)

5.268 std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference

Class [std::map](#) wrapper with performance instrumentation.

Inherits map< _Key, _Tp, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**

- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const_Key, _Tp >` **value_type**

Public Member Functions

- **map** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
map (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **map** (const `_Base` &__x)
- **map** (`map` &&__x)
- **map** (const `map` &__x)
- **map** (`initializer_list`< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- `mapped_type` & **at** (const `key_type` &__k)
- const `mapped_type` & **at** (const `key_type` &__k) const
- `iterator` **begin** ()
- const `iterator` **begin** () const
- const `iterator` **cbegin** () const
- const `iterator` **cend** () const
- void **clear** ()
- `size_type` **count** (const `key_type` &__x) const
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- const `iterator` **end** () const
- `std::pair`< const `iterator`, const `iterator` > **equal_range** (const `key_type` &__x) const

- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__x)
- size_type **erase** (const key_type &__x)
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- [std::pair](#)< iterator, bool > **insert** (const value_type &__x)
- iterator **insert** (iterator __position, const value_type &__x)
- void **insert** ([std::initializer_list](#)< value_type > __list)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- [map](#) & **operator=** (const [map](#) &__x)
- [map](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [map](#) & **operator=** ([map](#) &&__x)
- mapped_type & **operator**[] (const key_type &__k)
- [const_reverse_iterator](#) **rbegin** () const
- [reverse_iterator](#) **rbegin** ()
- [reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- void **swap** ([map](#) &__x)
- const_iterator **upper_bound** (const key_type &__x) const
- iterator **upper_bound** (const key_type &__x)

5.268.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::_profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class [std::map](#) wrapper with performance instrumentation.

Definition at line 47 of file profile/map.h.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

5.269 `std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::multimap` wrapper with performance instrumentation.

Inherits `multimap< _Key, _Tp, _Compare, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **multimap** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator` >
multimap (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multimap** (`const _Base &__x`)
- **multimap** (`multimap &&__x`)
- **multimap** (`const multimap &__x`)
- **multimap** (`initializer_list< value_type > __l, const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- iterator **begin** ()
- `const_iterator` **begin** () const

- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- iterator **end** ()
- const_iterator **end** () const
- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__x)
- [std::pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const [value_type](#) &__x)
- void **insert** ([std::initializer_list](#)< [value_type](#) > __list)
- iterator **insert** (iterator __position, const [value_type](#) &__x)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- [multimap](#) & **operator=** (const [multimap](#) &__x)
- [multimap](#) & **operator=** ([initializer_list](#)< [value_type](#) > __l)
- [multimap](#) & **operator=** ([multimap](#) &&__x)
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rbegin** ()
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- void **swap** ([multimap](#) &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

5.269.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class [std::multimap](#) wrapper with performance instrumentation.

Definition at line 41 of file profile/multimap.h.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

5.270 **std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference**

Class [std::multiset](#) wrapper with performance instrumentation.

Inherits [multiset< _Key, _Compare, _Allocator >](#).

Public Types

- typedef [_Allocator](#) **allocator_type**
- typedef [_Base::const_iterator](#) **const_iterator**
- typedef [_Base::const_pointer](#) **const_pointer**
- typedef [_Base::const_reference](#) **const_reference**
- typedef [_Base::const_reverse_iterator](#) **const_reverse_iterator**
- typedef [_Base::difference_type](#) **difference_type**
- typedef [_Base::iterator](#) **iterator**
- typedef [_Compare](#) **key_compare**
- typedef [_Key](#) **key_type**
- typedef [_Base::pointer](#) **pointer**
- typedef [_Base::reference](#) **reference**
- typedef [_Base::reverse_iterator](#) **reverse_iterator**
- typedef [_Base::size_type](#) **size_type**
- typedef [_Compare](#) **value_compare**
- typedef [_Key](#) **value_type**

Public Member Functions

- **multiset** (const [_Compare](#) &__comp=[_Compare](#)(), const [_Allocator](#) &__a=[_Allocator](#)())
- template<typename [_InputIterator](#) >
multiset ([_InputIterator](#) __first, [_InputIterator](#) __last, const [_Compare](#) &__comp=[_Compare](#)(), const [_Allocator](#) &__a=[_Allocator](#)())
- **multiset** (const [_Base](#) &__x)
- **multiset** ([multiset](#) &&__x)
- **multiset** (const [multiset](#) &__x)
- **multiset** ([initializer_list](#)< [value_type](#) > __l, const [_Compare](#) &__comp=[_Compare](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- [_Base](#) & **_M_base** ()
- const [_Base](#) & **_M_base** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const

- `const_iterator` **cend** () const
- `void` **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x) const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)
- `const_iterator` **find** (const `key_type` &__x) const
- `iterator` **insert** (const `value_type` &__x)
- `iterator` **insert** (`iterator` __position, const `value_type` &__x)
- `template`< `typename` `_InputIterator` >
`void` **insert** (`_InputIterator` __first, `_InputIterator` __last)
- `void` **insert** (`initializer_list`< `value_type` > __l)
- `iterator` **lower_bound** (const `key_type` &__x)
- `const_iterator` **lower_bound** (const `key_type` &__x) const
- `multiset` & **operator=** (const `multiset` &__x)
- `multiset` & **operator=** (`initializer_list`< `value_type` > __l)
- `multiset` & **operator=** (`multiset` &&__x)
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () const
- `void` **swap** (`multiset` &__x)
- `iterator` **upper_bound** (const `key_type` &__x)
- `const_iterator` **upper_bound** (const `key_type` &__x) const

5.270.1 Detailed Description

`template`< `typename` `_Key`, `typename` `_Compare` = `std::less`< `_Key`>, `typename` `_Allocator` = `std::allocator`< `_Key`>> `class` `std::__profile::multiset`< `_Key`, `_Compare`, `_Allocator` >

Class `std::multiset` wrapper with performance instrumentation.

Definition at line 41 of file `profile/multiset.h`.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

5.271 `std::__profile::set<_Key, _Compare, _Allocator >` Class Template Reference

Class `std::set` wrapper with performance instrumentation.

Inherits `set<_Key, _Compare, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **set** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
 set (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **set** (const `_Base` &__x)
- **set** (`set` &&__x)
- **set** (const `set` &__x)
- **set** (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const

- `const_iterator` **cend** () const
- `void` **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x) const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)
- `const_iterator` **find** (const `key_type` &__x) const
- `std::pair`< `iterator`, `bool` > **insert** (const `value_type` &__x)
- `iterator` **insert** (`iterator` __position, const `value_type` &__x)
- `template`< `typename` `_InputIterator` >
 `void` **insert** (`_InputIterator` __first, `_InputIterator` __last)
- `void` **insert** (`initializer_list`< `value_type` > __l)
- `iterator` **lower_bound** (const `key_type` &__x)
- `const_iterator` **lower_bound** (const `key_type` &__x) const
- `set` & **operator=** (const `set` &__x)
- `set` & **operator=** (`initializer_list`< `value_type` > __l)
- `set` & **operator=** (`set` &&__x)
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () const
- `void` **swap** (`set` &__x)
- `iterator` **upper_bound** (const `key_type` &__x)
- `const_iterator` **upper_bound** (const `key_type` &__x) const

5.271.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _
Allocator = std::allocator<_Key>> class std::__profile::set< _Key, _Compare,
_Allocator >
```

Class `std::set` wrapper with performance instrumentation.

Definition at line 41 of file `profile/set.h`.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

5.272 **std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference**

Class [std::unordered_map](#) wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::mapped_type` **mapped_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_map** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`
unordered_map (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_map** ([unordered_map](#) &&__x)
- **unordered_map** ([initializer_list](#)< `value_type` > __l, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_map** (`const _Base &__x`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void clear ()`
- `template<typename _InputIter >`
`void insert (_InputIter __first, _InputIter __last)`
- `void insert (const value_type * __first, const value_type * __last)`
- `iterator insert (const_iterator __iter, const value_type &__v)`

- void **insert** ([std::initializer_list](#)< value_type > __l)
- [std::pair](#)< iterator, bool > **insert** (const value_type &__obj)
- [unordered_map](#) & **operator=** ([unordered_map](#) &&__x)
- [unordered_map](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_map](#) & **operator=** (const [unordered_map](#) &__x)
- mapped_type & **operator[]** (const _Key &_k)
- void **rehash** (size_type __n)
- void **swap** ([unordered_map](#) &__x)

5.272.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>
```

Class [std::unordered_map](#) wrapper with performance instrumentation.

Definition at line 56 of file profile/unordered_map.

The documentation for this class was generated from the following file:

- [profile/unordered_map](#)

5.273 [std::__profile::unordered_multimap](#)< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Class [std::unordered_multimap](#) wrapper with performance instrumentation.

Inherits [_GLIBCXX_STD_PR::unordered_multimap](#)< _Key, _Tp, _Hash, _Pred, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**

- typedef **_Base::key_type** **key_type**
- typedef **_Base::reference** **reference**
- typedef **_Base::size_type** **size_type**
- typedef **_Base::value_type** **value_type**

Public Member Functions

- **unordered_multimap** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multimap (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** ([unordered_multimap](#) &&__x)
- **unordered_multimap** ([initializer_list](#)< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (const **_Base** &__x)
- **_Base** & **_M_base** ()
- const **_Base** & **_M_base** () const
- void **clear** ()
- void **insert** (const value_type *__first, const value_type *__last)
- template<typename _InputIter >
void **insert** (_InputIter __first, _InputIter __last)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- void **insert** ([std::initializer_list](#)< value_type > __l)
- iterator **insert** (const value_type &__obj)
- [unordered_multimap](#) & **operator=** ([unordered_multimap](#) &&__x)
- [unordered_multimap](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_multimap](#) & **operator=** (const [unordered_multimap](#) &__x)
- void **rehash** (size_type __n)
- void **swap** ([unordered_multimap](#) &__x)

5.273.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class [std::unordered_multimap](#) wrapper with performance instrumentation.

Definition at line 296 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered_map](#)

5.274 **std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference**

Unordered_multiset wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multiset (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** ([unordered_multiset](#) &&__x)
- **unordered_multiset** ([initializer_list](#)< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const _Base &__x)
- void **clear** ()
- template<typename _InputIter >
void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type * __first, const value_type * __last)
- void **insert** ([std::initializer_list](#)< value_type > __l)

- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- [unordered_multiset](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_multiset](#) & **operator=** ([unordered_multiset](#) &&__x)
- [unordered_multiset](#) & **operator=** (const [unordered_multiset](#) &__x)
- void **rehash** (size_type __n)
- void **swap** ([unordered_multiset](#) &__x)

5.274.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Unordered_multiset wrapper with performance instrumentation.

Definition at line 284 of file profile/unordered_set.

The documentation for this class was generated from the following file:

- [profile/unordered_set](#)

5.275 **std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference**

Unordered_set wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_set (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** ([unordered_set](#) &&__x)
- **unordered_set** ([initializer_list](#)< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (const _Base &__x)
- void **clear** ()
- template<typename _InputIter >
void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type *__first, const value_type *__last)
- void **insert** ([std::initializer_list](#)< value_type > __l)
- [std::pair](#)< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- [unordered_set](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_set](#) & **operator=** ([unordered_set](#) &&__x)
- [unordered_set](#) & **operator=** (const [unordered_set](#) &__x)
- void **rehash** (size_type __n)
- void **swap** ([unordered_set](#) &__x)

5.275.1 Detailed Description

```
template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class
std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >
```

Unordered_set wrapper with performance instrumentation.

Definition at line 56 of file profile/unordered_set.

The documentation for this class was generated from the following file:

- [profile/unordered_set](#)

5.276 std::_Base_bitset< _Nw > Struct Template Reference

Inheritance diagram for std::_Base_bitset< _Nw >:



Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- **_Base_bitset** (unsigned long long __val)
- size_t **_M_are_all_aux** () const
- void **_M_do_and** (const [_Base_bitset](#)< _Nw > &__x)
- size_t **_M_do_count** () const
- size_t **_M_do_find_first** (size_t __not_found) const
- size_t **_M_do_find_next** (size_t __prev, size_t __not_found) const
- void **_M_do_flip** ()
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const [_Base_bitset](#)< _Nw > &__x)
- void **_M_do_reset** ()
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const [_Base_bitset](#)< _Nw > &__x)
- const _WordT * **_M_getdata** () const
- _WordT **_M_getword** (size_t __pos) const
- _WordT & **_M_getword** (size_t __pos)
- _WordT **_M_hiword** () const
- _WordT & **_M_hiword** ()
- bool **_M_is_any** () const
- bool **_M_is_equal** (const [_Base_bitset](#)< _Nw > &__x) const

Static Public Member Functions

- static `_WordT _S_maskbit` (`size_t __pos`)
- static `size_t _S_whichbit` (`size_t __pos`)
- static `size_t _S_whichbyte` (`size_t __pos`)
- static `size_t _S_whichword` (`size_t __pos`)

Public Attributes

- `_WordT _M_w` [`_Nw`]

5.276.1 Detailed Description

`template<size_t _Nw> struct std::_Base_bitset< _Nw >`

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.

See documentation for `bitset`.

Definition at line 68 of file `bitset`.

5.276.2 Member Data Documentation

5.276.2.1 `template<size_t _Nw> _WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

Definition at line 73 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.277 `std::_Base_bitset< 0 >` Struct Template Reference

Public Types

- typedef unsigned long `_WordT`

Public Member Functions

- **_Base_bitset** (unsigned long long)
- **size_t _M_are_all_aux** () const
- **void _M_do_and** (const [_Base_bitset](#)< 0 > &)
- **size_t _M_do_count** () const
- **size_t _M_do_find_first** (size_t) const
- **size_t _M_do_find_next** (size_t, size_t) const
- **void _M_do_flip** ()
- **void _M_do_left_shift** (size_t)
- **void _M_do_or** (const [_Base_bitset](#)< 0 > &)
- **void _M_do_reset** ()
- **void _M_do_right_shift** (size_t)
- **void _M_do_set** ()
- **unsigned long long _M_do_to_ullong** () const
- **unsigned long _M_do_to_ulong** () const
- **void _M_do_xor** (const [_Base_bitset](#)< 0 > &)
- **_WordT & _M_getword** (size_t) const
- **_WordT _M_hiword** () const
- **bool _M_is_any** () const
- **bool _M_is_equal** (const [_Base_bitset](#)< 0 > &) const

Static Public Member Functions

- **static _WordT _S_maskbit** (size_t __pos)
- **static size_t _S_whichbit** (size_t __pos)
- **static size_t _S_whichbyte** (size_t __pos)
- **static size_t _S_whichword** (size_t __pos)

5.277.1 Detailed Description

template<> struct std::_Base_bitset< 0 >

Base class, specialization for no storage (zero-length bitset).

See documentation for bitset.

Definition at line 510 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.278 `std::_Base_bitset< 1 >` Struct Template Reference

Public Types

- typedef unsigned long `_WordT`

Public Member Functions

- `_Base_bitset` (unsigned long long `__val`)
- `size_t _M_are_all_aux` () const
- `void _M_do_and` (const `_Base_bitset< 1 >` &`__x`)
- `size_t _M_do_count` () const
- `size_t _M_do_find_first` (size_t `__not_found`) const
- `size_t _M_do_find_next` (size_t `__prev`, size_t `__not_found`) const
- `void _M_do_flip` ()
- `void _M_do_left_shift` (size_t `__shift`)
- `void _M_do_or` (const `_Base_bitset< 1 >` &`__x`)
- `void _M_do_reset` ()
- `void _M_do_right_shift` (size_t `__shift`)
- `void _M_do_set` ()
- unsigned long long `_M_do_to_ullong` () const
- unsigned long `_M_do_to_ulong` () const
- `void _M_do_xor` (const `_Base_bitset< 1 >` &`__x`)
- const `_WordT * _M_getdata` () const
- `_WordT _M_getword` (size_t) const
- `_WordT & _M_getword` (size_t)
- `_WordT _M_hiword` () const
- `_WordT & _M_hiword` ()
- `bool _M_is_any` () const
- `bool _M_is_equal` (const `_Base_bitset< 1 >` &`__x`) const

Static Public Member Functions

- static `_WordT _S_maskbit` (size_t `__pos`)
- static `size_t _S_whichbit` (size_t `__pos`)
- static `size_t _S_whichbyte` (size_t `__pos`)
- static `size_t _S_whichword` (size_t `__pos`)

Public Attributes

- `_WordT _M_w`

5.278.1 Detailed Description

`template<> struct std::_Base_bitset< 1 >`

Base class, specialization for a single word.

See documentation for `bitset`.

Definition at line 366 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.279 `std::_Build_index_tuple< _Num >` Struct Template Reference

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

Public Types

- `typedef _Build_index_tuple< _Num-1 >::__type::__next __type`

5.279.1 Detailed Description

`template<std::size_t _Num> struct std::_Build_index_tuple< _Num >`

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

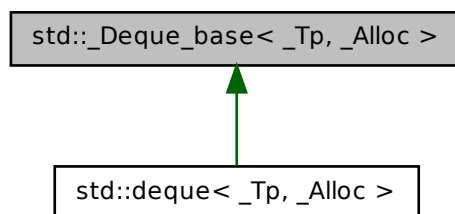
Definition at line 711 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.280 `std::_Deque_base< _Tp, _Alloc >` Class Template Reference

Inheritance diagram for `std::_Deque_base< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Deque_iterator< _Tp, const _Tp &, const _Tp * >` **const_iterator**
- typedef `_Deque_iterator< _Tp, _Tp &, _Tp * >` **iterator**

Public Member Functions

- `_Deque_base` (`const allocator_type &__a, size_t __num_elements`)
- `_Deque_base` (`_Deque_base &&__x`)
- `_Deque_base` (`const allocator_type &__a`)
- `_Deque_base` (`size_t __num_elements`)
- `allocator_type` **get_allocator** () const

Protected Types

- enum { `_S_initial_map_size` }
- typedef `_Alloc::template rebind< _Tp * >::other` **_Map_alloc_type**
- typedef `_Alloc::template rebind< _Tp >::other` **_Tp_alloc_type**

Protected Member Functions

- `_Tp** _M_allocate_map (size_t __n)`
- `_Tp* _M_allocate_node ()`
- `void _M_create_nodes (_Tp** __nstart, _Tp** __nfinish)`
- `void _M_deallocate_map (_Tp** __p, size_t __n)`
- `void _M_deallocate_node (_Tp* __p)`
- `void _M_destroy_nodes (_Tp** __nstart, _Tp** __nfinish)`
- `_Map_alloc_type _M_get_map_allocator () const`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `void _M_initialize_map (size_t)`

Protected Attributes

- `_Deque_impl _M_impl`

5.280.1 Detailed Description

`template<typename _Tp, typename _Alloc> class std::_Deque_base< _Tp, _Alloc >`

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual `Tp` element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 436 of file `stl_deque.h`.

5.280.2 Member Function Documentation

5.280.2.1 `template<typename _Tp , typename _Alloc > void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t __num_elements) [protected]`

Layout storage.

Parameters

num_elements The count of `T`'s for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 572 of file `stl_deque.h`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_range_initialize()`.

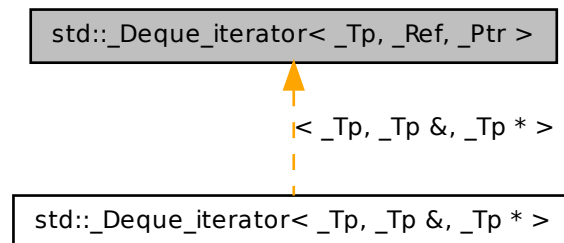
The documentation for this class was generated from the following file:

- [stl_deque.h](#)

5.281 `std::_Deque_iterator< _Tp, _Ref, _Ptr >` Struct Template Reference

A `deque::iterator`.

Inheritance diagram for `std::_Deque_iterator< _Tp, _Ref, _Ptr >`:

**Public Types**

- `typedef _Tp ** _Map_pointer`
- `typedef _Deque_iterator _Self`
- `typedef _Deque_iterator< _Tp, const _Tp &, const _Tp * > const_iterator`
- `typedef ptrdiff_t difference_type`

- typedef [_Deque_iterator](#)< _Tp, _Tp &, _Tp * > **iterator**
- typedef [std::random_access_iterator_tag](#) **iterator_category**
- typedef _Ptr **pointer**
- typedef _Ref **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [_Deque_iterator](#) (_Tp *__x, _Map_pointer __y)
- [_Deque_iterator](#) (const [iterator](#) &__x)
- void [_M_set_node](#) (_Map_pointer __new_node)
- reference **operator*** () const
- [_Self](#) **operator+** (difference_type __n) const
- [_Self](#) **operator++** (int)
- [_Self](#) & **operator++** ()
- [_Self](#) & **operator+=** (difference_type __n)
- [_Self](#) **operator-** (difference_type __n) const
- [_Self](#) & **operator--** ()
- [_Self](#) **operator--** (int)
- [_Self](#) & **operator-=** (difference_type __n)
- pointer **operator->** () const
- reference **operator[]** (difference_type __n) const

Static Public Member Functions

- static size_t [_S_buffer_size](#) ()

Public Attributes

- _Tp * [_M_cur](#)
- _Tp * [_M_first](#)
- _Tp * [_M_last](#)
- _Map_pointer [_M_node](#)

5.281.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct std::_Deque_iterator< _Tp, _Ref, _Ptr >

A deque::iterator. Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its

valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 103 of file `stl_deque.h`.

5.281.2 Member Function Documentation

5.281.2.1 `template<typename _Tp, typename _Ref, typename _Ptr> void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (_Map_pointer __new_node) [inline]`

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

Definition at line 231 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl_deque.h](#)

5.282 `std::_Derives_from_binary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [binary_function](#).

Inherits `__sfinae_types`.

Static Public Attributes

- static const bool `value`

5.282.1 Detailed Description

`template<typename _Tp> struct std::_Derives_from_binary_function< _Tp >`

Determines if the type `_Tp` derives from [binary_function](#).

Definition at line 190 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.283 `std::_Derives_from_unary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [unary_function](#).

Inherits `__sfinae_types`.

Static Public Attributes

- static const bool **value**

5.283.1 Detailed Description

```
template<typename _Tp> struct std::_Derives_from_unary_function< _Tp >
```

Determines if the type `_Tp` derives from [unary_function](#).

Definition at line 174 of file `functional`.

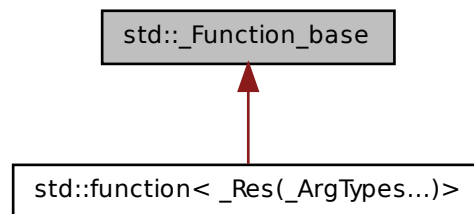
The documentation for this struct was generated from the following file:

- [functional](#)

5.284 `std::_Function_base` Class Reference

Base class of all polymorphic function object wrappers.

Inheritance diagram for `std::_Function_base`:



Public Types

- `typedef bool(* _Manager_type)(_Any_data &, const _Any_data &, _Manager_operation)`

Public Member Functions

- `bool _M_empty () const`

Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

Static Public Attributes

- `static const std::size_t _M_max_align`
- `static const std::size_t _M_max_size`

5.284.1 Detailed Description

Base class of all polymorphic function object wrappers.

Definition at line 1470 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.285 `std::_Function_to_function_pointer< _Tp, _IsFunctionType >` Struct Template Reference

Turns a function type into a function pointer type.

Public Types

- `typedef _Tp type`

5.285.1 Detailed Description

```
template<typename _Tp, bool _IsFunctionType = is_function<_Tp>::value>  
struct std::_Function_to_function_pointer< _Tp, _IsFunctionType >
```

Turns a function type into a function pointer type.

Definition at line 206 of file functional.

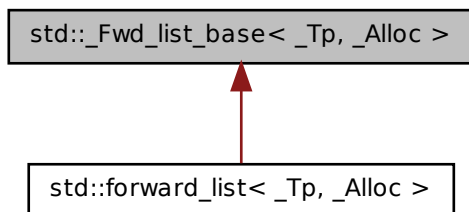
The documentation for this struct was generated from the following file:

- [functional](#)

5.286 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference

Base class for forward_list.

Inheritance diagram for std::_Fwd_list_base< _Tp, _Alloc >:



Public Types

- typedef [_Fwd_list_node](#)< _Tp > **_Node**
- typedef [_Fwd_list_const_iterator](#)< _Tp > **const_iterator**
- typedef [_Fwd_list_iterator](#)< _Tp > **iterator**

Public Member Functions

- [_Fwd_list_base](#) (const _Alloc &__a)

- [_Fwd_list_base](#) ([_Fwd_list_base](#) &&__lst)
- [_Fwd_list_base](#) (const [_Fwd_list_base](#) &__lst, const [_Alloc](#) &__a)
- [_Fwd_list_base](#) ([_Fwd_list_base](#) &&__lst, const [_Alloc](#) &__a)
- const [_Node_alloc_type](#) & [_M_get_Node_allocator](#) () const
- [_Node_alloc_type](#) & [_M_get_Node_allocator](#) ()

Protected Types

- typedef [_Alloc](#)::template rebind< [_Fwd_list_node](#)< [_Tp](#) > >::other [_Node_alloc_type](#)
- typedef [_Alloc](#)::template rebind< [_Tp](#) >::other [_Tp_alloc_type](#)

Protected Member Functions

- template<typename... [_Args](#)>
[_Node](#) * [_M_create_node](#) ([_Args](#) &&...__args)
- void [_M_erase_after](#) ([_Fwd_list_node_base](#) *__pos)
- void [_M_erase_after](#) ([_Fwd_list_node_base](#) *__pos, [_Fwd_list_node_base](#) *__last)
- [_Node](#) * [_M_get_node](#) ()
- template<typename... [_Args](#)>
[_Fwd_list_node_base](#) * [_M_insert_after](#) (const_iterator __pos, [_Args](#) &&...__args)
- void [_M_put_node](#) ([_Node](#) *__p)

Protected Attributes

- [_Fwd_list_impl](#) [_M_impl](#)

5.286.1 Detailed Description

template<typename [_Tp](#), typename [_Alloc](#)> struct std:: [_Fwd_list_base](#)< [_Tp](#), [_Alloc](#) >

Base class for forward_list.

Definition at line 274 of file forward_list.h.

The documentation for this struct was generated from the following files:

- [forward_list.h](#)
- [forward_list.tcc](#)

5.287 `std::_Fwd_list_const_iterator< _Tp >` Struct Template Reference

A `forward_list::const_iterator`.

Public Types

- typedef const [_Fwd_list_node](#)< `_Tp` > `_Node`
- typedef [_Fwd_list_const_iterator](#)< `_Tp` > `_Self`
- typedef `ptrdiff_t` `difference_type`
- typedef [_Fwd_list_iterator](#)< `_Tp` > `iterator`
- typedef [std::forward_iterator_tag](#) `iterator_category`
- typedef const `_Tp` * `pointer`
- typedef const `_Tp` & `reference`
- typedef `_Tp` `value_type`

Public Member Functions

- `_Fwd_list_const_iterator` (const [_Fwd_list_node_base](#) * __n)
- `_Fwd_list_const_iterator` (const [iterator](#) & __iter)
- `_Self _M_next` () const
- bool `operator!=` (const `_Self` & __x) const
- reference `operator*` () const
- `_Self operator++` (int)
- `_Self & operator++` ()
- pointer `operator->` () const
- bool `operator==` (const `_Self` & __x) const

Public Attributes

- const [_Fwd_list_node_base](#) * `_M_node`

5.287.1 Detailed Description

```
template<typename _Tp> struct std::_Fwd_list_const_iterator< _Tp >
```

A `forward_list::const_iterator`. All the functions are op overloads.

Definition at line 186 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

5.288 `std::_Fwd_list_iterator<_Tp>` Struct Template Reference

A `forward_list::iterator`.

Public Types

- typedef [`_Fwd_list_node`](#)<_Tp> **_Node**
- typedef [`_Fwd_list_iterator`](#)<_Tp> **_Self**
- typedef ptrdiff_t **difference_type**
- typedef [`std::forward_iterator_tag`](#) **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- [`_Fwd_list_iterator`](#) ([`_Fwd_list_node_base`](#) *__n)
- [`_Self`](#) [`_M_next`](#) () const
- bool **operator!=** (const [`_Self`](#) &__x) const
- reference **operator*** () const
- [`_Self`](#) & **operator++** ()
- [`_Self`](#) **operator++** (int)
- pointer **operator->** () const
- bool **operator==** (const [`_Self`](#) &__x) const

Public Attributes

- [`_Fwd_list_node_base`](#) * **_M_node**

5.288.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_iterator<_Tp>`

A `forward_list::iterator`. All the functions are op overloads.

Definition at line 118 of file `forward_list.h`.

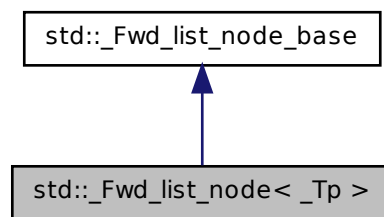
The documentation for this struct was generated from the following file:

- [`forward_list.h`](#)

5.289 std::_Fwd_list_node< _Tp > Struct Template Reference

A helper node class for forward_list. This is just a linked list with a data value in each node. There is a sorting utility method.

Inheritance diagram for std::_Fwd_list_node< _Tp >:



Public Member Functions

- `template<typename... _Args>`
`_Fwd_list_node` (`_Args &&...__args`)
- `void _M_reverse_after` ()
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin`, `_Fwd_list_node_base * __end`)
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin`)

Static Public Member Functions

- `static void swap` (`_Fwd_list_node_base & __x`, `_Fwd_list_node_base & __y`)

Public Attributes

- `_Fwd_list_node_base * _M_next`
- `_Tp _M_value`

5.289.1 Detailed Description

template<typename _Tp> struct std::_Fwd_list_node< _Tp >

A helper node class for forward_list. This is just a linked list with a data value in each node. There is a sorting utility method.

Definition at line 101 of file forward_list.h.

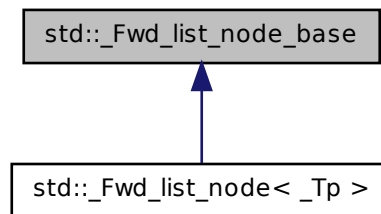
The documentation for this struct was generated from the following file:

- [forward_list.h](#)

5.290 std::_Fwd_list_node_base Struct Reference

A helper basic node class for forward_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Inheritance diagram for std::_Fwd_list_node_base:



Public Member Functions

- `void _M_reverse_after ()`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end)`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin)`

Static Public Member Functions

- static void `swap` (`_Fwd_list_node_base` &__x, `_Fwd_list_node_base` &__y)

Public Attributes

- `_Fwd_list_node_base` * `_M_next`

5.290.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Definition at line 44 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

5.291 `std::_Has_result_type_helper< _Tp >` Class Template Reference

Inherits `__sfinae_types`.

Static Public Attributes

- static const bool `value`

5.291.1 Detailed Description

`template<typename _Tp> class std::_Has_result_type_helper< _Tp >`

Actual implementation of `_Has_result_type`, which uses SFINAE to determine if the type `_Tp` has a publicly-accessible member type `result_type`.

Definition at line 72 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.292 `std::_Index_tuple< _Indexes >` Struct Template Reference

Public Types

- typedef [_Index_tuple](#)< _Indexes..., sizeof...(_Indexes)> `__next`

5.292.1 Detailed Description

`template<int... _Indexes> struct std::_Index_tuple< _Indexes >`

Stores a tuple of indices. Used by [bind\(\)](#) to extract the elements in a tuple.

Definition at line 704 of file `tuple`.

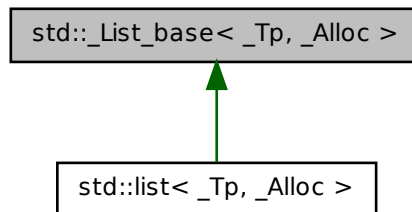
The documentation for this struct was generated from the following file:

- [tuple](#)

5.293 `std::_List_base< _Tp, _Alloc >` Class Template Reference

See [bits/stl_deque.h](#)'s [_Deque_base](#) for an explanation.

Inheritance diagram for `std::_List_base< _Tp, _Alloc >`:



Public Types

- typedef _Alloc **allocator_type**

Public Member Functions

- **_List_base** (const allocator_type &__a)
- **_List_base** ([_List_base](#) &&__x)
- void **_M_clear** ()
- const _Node_alloc_type & **_M_get_Node_allocator** () const
- _Node_alloc_type & **_M_get_Node_allocator** ()
- _Tp_alloc_type **_M_get_Tp_allocator** () const
- void **_M_init** ()
- allocator_type **get_allocator** () const

Protected Types

- typedef _Alloc::template rebind< [_List_node](#)< _Tp > ::other **_Node_alloc_type**
- typedef _Alloc::template rebind< _Tp > ::other **_Tp_alloc_type**

Protected Member Functions

- [_List_node](#)< _Tp > * **_M_get_node** ()
- void **_M_put_node** ([_List_node](#)< _Tp > *__p)

Protected Attributes

- [_List_impl](#) **_M_impl**

5.293.1 Detailed Description

template<typename _Tp, typename _Alloc> class std::_List_base< _Tp, _Alloc >

See [bits/stl_deque.h](#)'s [_Deque_base](#) for an explanation.

Definition at line 277 of file [stl_list.h](#).

The documentation for this class was generated from the following files:

- [stl_list.h](#)
- [list.tcc](#)

5.294 `std::_List_const_iterator< _Tp >` Struct Template Reference

A `list::const_iterator`.

Public Types

- typedef const `_List_node< _Tp > _Node`
- typedef `_List_const_iterator< _Tp > _Self`
- typedef `ptrdiff_t difference_type`
- typedef `_List_iterator< _Tp > iterator`
- typedef `std::bidirectional_iterator_tag iterator_category`
- typedef const `_Tp * pointer`
- typedef const `_Tp & reference`
- typedef `_Tp value_type`

Public Member Functions

- `_List_const_iterator` (const `_List_node_base * __x`)
- `_List_const_iterator` (const `iterator & __x`)
- bool `operator!=` (const `_Self & __x`) const
- reference `operator*` () const
- `_Self` `operator++` (int)
- `_Self & operator++` ()
- `_Self & operator--` ()
- `_Self` `operator--` (int)
- pointer `operator->` () const
- bool `operator==` (const `_Self & __x`) const

Public Attributes

- const `_List_node_base * _M_node`

5.294.1 Detailed Description

`template<typename _Tp> struct std::_List_const_iterator< _Tp >`

A `list::const_iterator`. All the functions are op overloads.

Definition at line 188 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.295 `std::_List_iterator<_Tp>` Struct Template Reference

A `list::iterator`.

Public Types

- typedef [_List_node](#)<_Tp> **_Node**
- typedef [_List_iterator](#)<_Tp> **_Self**
- typedef `ptrdiff_t` **difference_type**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- `_List_iterator` ([_List_node_base](#) *__x)
- `bool operator!=` (const [_Self](#) &__x) const
- `reference operator*` () const
- [_Self](#) & `operator++` ()
- [_Self](#) `operator++` (int)
- [_Self](#) & `operator--` ()
- [_Self](#) `operator--` (int)
- `pointer operator->` () const
- `bool operator==` (const [_Self](#) &__x) const

Public Attributes

- [_List_node_base](#) * **_M_node**

5.295.1 Detailed Description

`template<typename _Tp> struct std::_List_iterator<_Tp>`

A `list::iterator`. All the functions are op overloads.

Definition at line 113 of file `stl_list.h`.

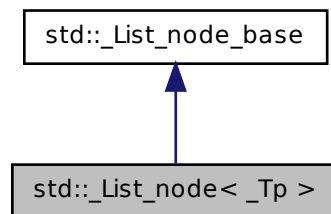
The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.296 `std::_List_node< _Tp >` Struct Template Reference

An actual node in the list.

Inheritance diagram for `std::_List_node< _Tp >`:



Public Member Functions

- `template<typename... _Args> _List_node (_Args &&...__args)`
- `void _M_hook (_List_node_base *const __position) throw ()`
- `void _M_reverse () throw ()`
- `void _M_transfer (_List_node_base *const __first, _List_node_base *const __last) throw ()`
- `void _M_unhook () throw ()`

Static Public Member Functions

- `static void swap (_List_node_base &__x, _List_node_base &__y) throw ()`

Public Attributes

- `_Tp _M_data`

- [_List_node_base](#) * [_M_next](#)
- [_List_node_base](#) * [_M_prev](#)

5.296.1 Detailed Description

`template<typename _Tp> struct std::_List_node< _Tp >`

An actual node in the list.

Definition at line 95 of file `stl_list.h`.

5.296.2 Member Data Documentation

5.296.2.1 `template<typename _Tp> _Tp std::_List_node< _Tp >::_M_data`

< User's data.

Definition at line 98 of file `stl_list.h`.

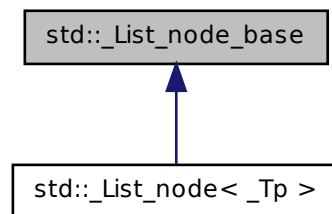
The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.297 std::_List_node_base Struct Reference

Common part of a node in the list.

Inheritance diagram for `std::_List_node_base`:



Public Member Functions

- void **_M_hook** ([_List_node_base](#) *const __position) throw ()
- void **_M_reverse** () throw ()
- void **_M_transfer** ([_List_node_base](#) *const __first, [_List_node_base](#) *const __last) throw ()
- void **_M_unhook** () throw ()

Static Public Member Functions

- static void **swap** ([_List_node_base](#) &__x, [_List_node_base](#) &__y) throw ()

Public Attributes

- [_List_node_base](#) * **_M_next**
- [_List_node_base](#) * **_M_prev**

5.297.1 Detailed Description

Common part of a node in the list.

Definition at line 71 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.298 `std::_Maybe_get_result_type< _Has_result_type, _Functor > Struct Template Reference`

If we have found a `result_type`, extract it.

Inheritance diagram for `std::_Maybe_get_result_type< _Has_result_type, _Functor >`:



5.298.1 Detailed Description

```
template<bool _Has_result_type, typename _Functor> struct std::_Maybe_get_
result_type< _Has_result_type, _Functor >
```

If we have found a result_type, extract it.

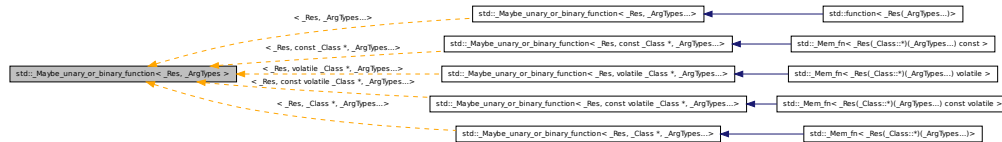
Definition at line 96 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.299 std::_Maybe_unary_or_binary_function< _Res, _ArgTypes > Struct Template Reference

Inheritance diagram for std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >:



5.299.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Maybe_unary_
or_binary_function< _Res, _ArgTypes >
```

Derives from [unary_function](#) or [binary_function](#), or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

Definition at line 470 of file functional.

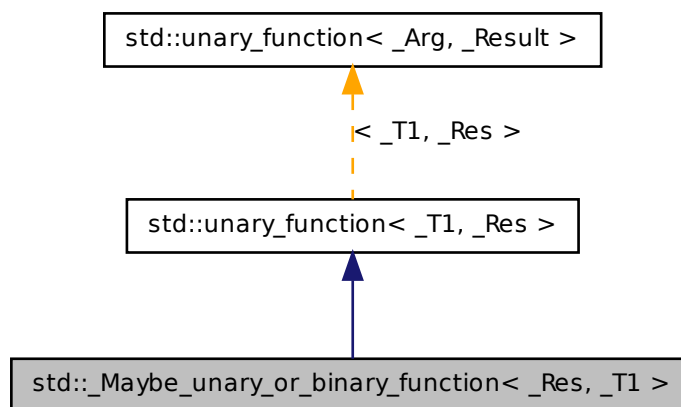
The documentation for this struct was generated from the following file:

- [functional](#)

5.300 `std::_Maybe_unary_or_binary_function< _Res, _T1 >` Struct Template Reference

Derives from `unary_function`, as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function< _Res, _T1 >`:



Public Types

- typedef `_T1` `argument_type`
- typedef `_Res` `result_type`

5.300.1 Detailed Description

```
template<typename _Res, typename _T1> struct std::_Maybe_unary_or_
binary_function< _Res, _T1 >
```

Derives from `unary_function`, as appropriate.

Definition at line 474 of file `functional`.

5.300.2 Member Typedef Documentation

5.300.2.1 typedef _T1 std::unary_function< _T1 , _Res >::argument_type **[inherited]**

argument_type is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.300.2.2 typedef _Res std::unary_function< _T1 , _Res >::result_type **[inherited]**

result_type is the return type

Definition at line 105 of file stl_function.h.

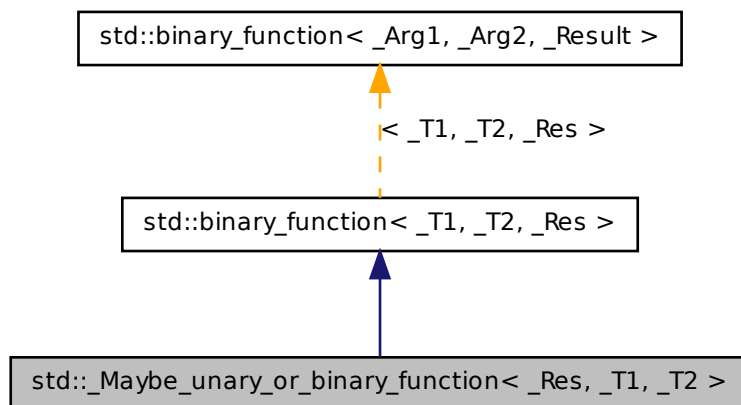
The documentation for this struct was generated from the following file:

- [functional](#)

5.301 std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 > Struct Template Reference

Derives from [binary_function](#), as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `_Res` [result_type](#)
- typedef `_T2` [second_argument_type](#)

5.301.1 Detailed Description

`template<typename _Res, typename _T1, typename _T2> struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`

Derives from [binary_function](#), as appropriate.

Definition at line 479 of file `functional`.

5.302 std::_Maybe_wrap_member_pointer< _Tp > Struct Template Reference

5.301.2 Member Typedef Documentation

5.301.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , _Res
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.301.2.2 `typedef _Res std::binary_function< _T1 , _T2 , _Res >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.301.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , _Res
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [functional](#)

5.302 std::_Maybe_wrap_member_pointer< _Tp > Struct Template Reference

Public Types

- `typedef _Tp type`

Static Public Member Functions

- `static const _Tp & __do_wrap (const _Tp &__x)`

5.302.1 Detailed Description

```
template<typename _Tp> struct std::_Maybe_wrap_member_pointer< _Tp >
```

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. The primary template handles the non--member-pointer case.

Definition at line 1027 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.303 `std::_Maybe_wrap_member_pointer< _Tp _Class::* >` Struct Template Reference

Public Types

- `typedef _Mem_fn< _Tp _Class::* > type`

Static Public Member Functions

- static type `__do_wrap (_Tp _Class::* __pm)`

5.303.1 Detailed Description

```
template<typename _Tp, typename _Class> struct std::_Maybe_wrap_member_pointer< _Tp _Class::* >
```

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. This partial specialization handles the member pointer case.

Definition at line 1042 of file `functional`.

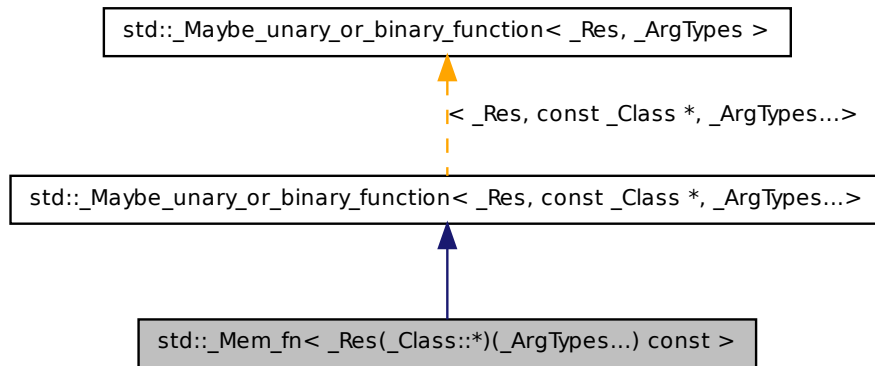
The documentation for this struct was generated from the following file:

- [functional](#)

5.304 `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >` Class Template Reference

Implementation of `mem_fn` for const member function pointers.

Inheritance diagram for `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >`:



Public Types

- `typedef _Res result_type`

Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (const _Class *__object, _ArgTypes... __args) const`
- `_Res operator() (const _Class &__object, _ArgTypes... __args) const`

5.304.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> class
std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >
```

Implementation of `mem_fn` for const member function pointers.

Definition at line 530 of file `functional`.

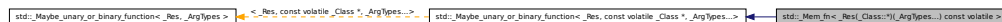
The documentation for this class was generated from the following file:

- [functional](#)

5.305 std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile > Class Template Reference

Implementation of `mem_fn` for const volatile member function pointers.

Inheritance diagram for `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >`:



Public Types

- typedef `_Res` **result_type**

Public Member Functions

- **_Mem_fn** (`_Functor __pmf`)
- template<typename `_Tp` >
`_Res operator()` (`_Tp &__object, _ArgTypes...__args`) const
- `_Res operator()` (`const volatile _Class *__object, _ArgTypes...__args`) const
- `_Res operator()` (`const volatile _Class &__object, _ArgTypes...__args`) const

5.305.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> class
std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >
```

Implementation of `mem_fn` for const volatile member function pointers.

5.306 `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference 1455

Definition at line 623 of file functional.

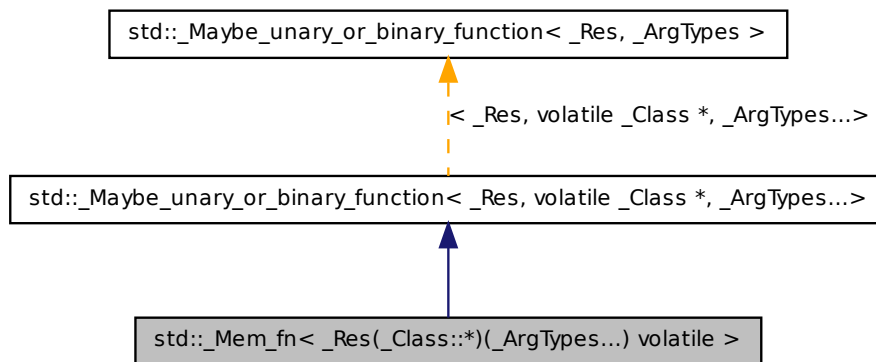
The documentation for this class was generated from the following file:

- [functional](#)

5.306 `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference

Implementation of `mem_fn` for volatile member function pointers.

Inheritance diagram for `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >`:



Public Types

- `typedef _Res result_type`

Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes...__args) const`
- `_Res operator() (volatile _Class *__object, _ArgTypes...__args) const`
- `_Res operator() (volatile _Class &__object, _ArgTypes...__args) const`

5.306.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> class
std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >
```

Implementation of `mem_fn` for volatile member function pointers.

Definition at line 576 of file `functional`.

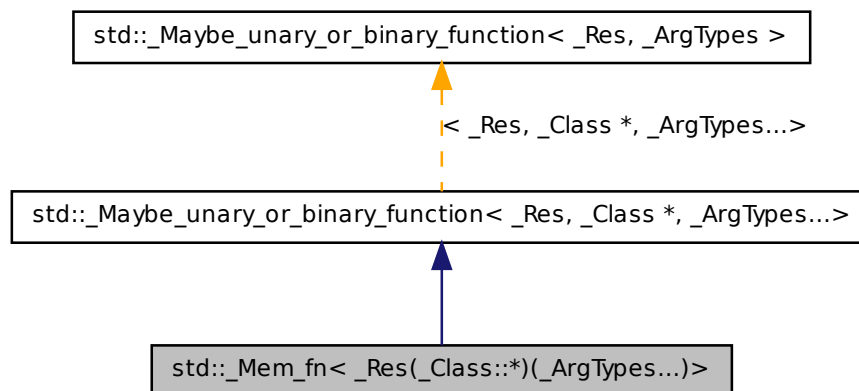
The documentation for this class was generated from the following file:

- [functional](#)

5.307 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>` Class Template Reference

Implementation of `mem_fn` for member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`:



Public Types

- `typedef _Res result_type`

Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (_Class *__object, _ArgTypes... __args) const`
- `_Res operator() (_Class &__object, _ArgTypes... __args) const`

5.307.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class std::_Mem_fn< _Res(_Class::*)(_ArgTypes...)>`

Implementation of `mem_fn` for member function pointers.

Definition at line 484 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.308 `std::_Mu< _Arg, false, false >` Class Template Reference

Public Member Functions

- `template<typename _CVArg, typename _Tuple > _CVArg && operator() (_CVArg &&__arg, _Tuple &&) const volatile`

5.308.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, false, false >`

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are the same as the cv-qualifiers on the `_Mu` object. [TR1 3.6.3/5 bullet 4]

Definition at line 1003 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.309 `std::_Mu< _Arg, false, true >` Class Template Reference

Public Member Functions

- `template<typename _Tuple >
result< _Mu(_Arg, _Tuple)>::type operator() (const volatile _Arg &, _Tuple
&&__tuple) const volatile`

5.309.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, false, true >`

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. [TR1 3.6.3/5 bullet 3]

Definition at line 969 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

5.310 `std::_Mu< _Arg, true, false >` Class Template Reference

Public Member Functions

- `template<typename _CVArg, typename... _Args>
result_of< _CVArg(_Args...)>::type operator() (_CVArg &__arg, tuple< _-
Args...> &&__tuple) const volatile`

5.310.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, true, false >`

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). [TR1 3.6.3/5 bullet 2]

Definition at line 928 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

5.311 std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference

Public Types

- typedef _Tp & **result_type**

Public Member Functions

- template<typename _CVRef, typename _Tuple >
result_type **operator()** (_CVRef &__arg, _Tuple &&) const volatile

5.311.1 Detailed Description

template<typename _Tp> class std::_Mu< reference_wrapper< _Tp >, false, false >

If the argument is reference_wrapper<_Tp>, returns the underlying reference. [TR1 3.6.3/5 bullet 1]

Definition at line 907 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

5.312 std::_Placeholder< _Num > Struct Template Reference

The type of placeholder objects defined by libstdc++.

5.312.1 Detailed Description

template<int _Num> struct std::_Placeholder< _Num >

The type of placeholder objects defined by libstdc++.

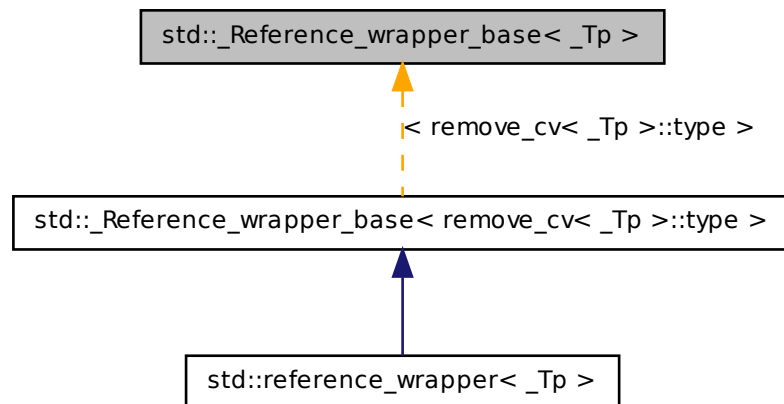
Definition at line 794 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.313 std::_Reference_wrapper_base< _Tp > Struct Template Reference

Inheritance diagram for std::_Reference_wrapper_base< _Tp >:



5.313.1 Detailed Description

template<typename _Tp> struct std::_Reference_wrapper_base< _Tp >

Derives from [unary_function](#) or [binary_function](#) when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both [unary_function](#) and [binary_function](#).

Definition at line 296 of file `functional`.

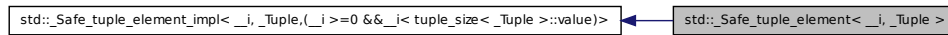
The documentation for this struct was generated from the following file:

- [functional](#)

5.314 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference 1461

5.314 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element< __i, _Tuple >`:



5.314.1 Detailed Description

template<int __i, typename _Tuple> struct `std::_Safe_tuple_element< __i, _Tuple >`

Like `tuple_element`, but returns `_No_tuple_element` when `tuple_element` would return an error.

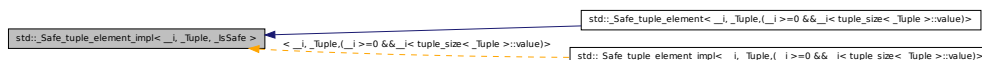
Definition at line 881 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.315 `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`:



5.315.1 Detailed Description

```
template<int __i, typename _Tuple, bool _IsSafe> struct std::_Safe_tuple_ -  
element_impl< __i, _Tuple, _IsSafe >
```

Implementation helper for [_Safe_tuple_element](#). This primary template handles the case where it is safe to use `tuple_element`.

Definition at line 862 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.316 `std::_Safe_tuple_element_impl< __i, _Tuple, false >` Struct Template Reference

Public Types

- `typedef _No_tuple_element type`

5.316.1 Detailed Description

```
template<int __i, typename _Tuple> struct std::_Safe_tuple_element_impl< __i,  
_Tuple, false >
```

Implementation helper for [_Safe_tuple_element](#). This partial specialization handles the case where it is not safe to use `tuple_element`. We just return `_No_tuple_element`.

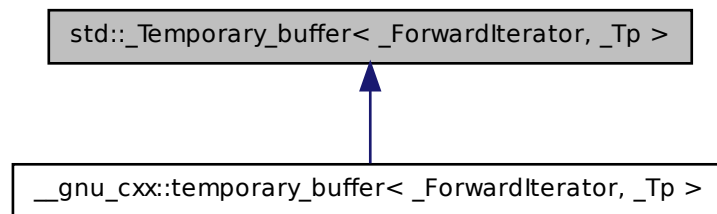
Definition at line 871 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.317 `std::_Temporary_buffer< _ForwardIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::_Temporary_buffer< _ForwardIterator, _Tp >`:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [_Temporary_buffer](#) (_ForwardIterator __first, _ForwardIterator __last)
- iterator [begin](#) ()
- iterator [end](#) ()
- size_type [requested_size](#) () const
- size_type [size](#) () const

Protected Attributes

- pointer **_M_buffer**
- size_type **_M_len**
- size_type **_M_original_len**

5.317.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp> class std::_Temporary_-  
buffer< _ForwardIterator, _Tp >
```

This class is used in two places: [stl_algo.h](#) and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 121 of file `stl_tempbuf.h`.

5.317.2 Constructor & Destructor Documentation

```
5.317.2.1 template<typename _ForwardIterator , typename _Tp  
> std::_Temporary_buffer< _ForwardIterator, _Tp  
>::_Temporary_buffer ( _ForwardIterator __first,  
_ForwardIterator __last )
```

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file `stl_tempbuf.h`.

References `std::pair< _T1, _T2 >::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair< _T1, _T2 >::second`.

5.317.3 Member Function Documentation

```
5.317.3.1 template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( )  
[inline]
```

As per Table mumble.

Definition at line 150 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

```
5.317.3.2 template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( )  
[inline]
```

As per Table mumble.

Definition at line 155 of file `stl_tempbuf.h`.

5.317.3.3 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size () const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 145 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

5.317.3.4 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline]`

As per Table mumble.

Definition at line 140 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this class was generated from the following file:

- [stl_tempbuf.h](#)

5.318 `std::_Tuple_impl< _Idx >` Struct Template Reference

Protected Member Functions

- `void _M_swap_impl (_Tuple_impl &)`

5.318.1 Detailed Description

`template<std::size_t _Idx> struct std::_Tuple_impl< _Idx >`

Zero-element tuple implementation. This is the basis case for the inheritance recursion.

Definition at line 125 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.319 `std::_Tuple_impl<_Idx, _Head, _Tail...>` Struct Template Reference

Inherits `_Tuple_impl<_Idx+1, _Tail...>`, and `_Head_base<_Idx, _Head, std::is_empty<_Head>::value>`.

Public Types

- `typedef _Head_base<_Idx, _Head, std::is_empty<_Head>::value> _Base`
- `typedef _Tuple_impl<_Idx+1, _Tail...> _Inherited`

Public Member Functions

- `_Tuple_impl` (const `_Tuple_impl` &__in)
- `_Tuple_impl` (`_Tuple_impl` &&__in)
- `template<typename... _UElements>`
`_Tuple_impl` (const `_Tuple_impl<_Idx, _UElements...>` &__in)
- `template<typename... _UElements>`
`_Tuple_impl` (`_Tuple_impl<_Idx, _UElements...>` &&__in)
- `_Tuple_impl` (const `_Head` &__head, const `_Tail` &...__tail)
- `template<typename _UHead, typename... _UTail>`
`_Tuple_impl` (`_UHead` &&__head, `_UTail` &&...__tail)
- `_Head` & `_M_head` ()
- const `_Head` & `_M_head` () const
- const `_Inherited` & `_M_tail` () const
- `_Inherited` & `_M_tail` ()
- `template<typename... _UElements>`
`_Tuple_impl` & `operator=` (`_Tuple_impl<_Idx, _UElements...>` &&__in)
- `_Tuple_impl` & `operator=` (const `_Tuple_impl` &__in)
- `_Tuple_impl` & `operator=` (`_Tuple_impl` &&__in)
- `template<typename... _UElements>`
`_Tuple_impl` & `operator=` (const `_Tuple_impl<_Idx, _UElements...>` &__in)

Protected Member Functions

- void `_M_swap_impl` (`_Tuple_impl` &__in)

5.319.1 Detailed Description

template<std::size_t _Idx, typename _Head, typename... _Tail> struct std::_Tuple_impl< _Idx, _Head, _Tail...>

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

Definition at line 137 of file `tuple`.

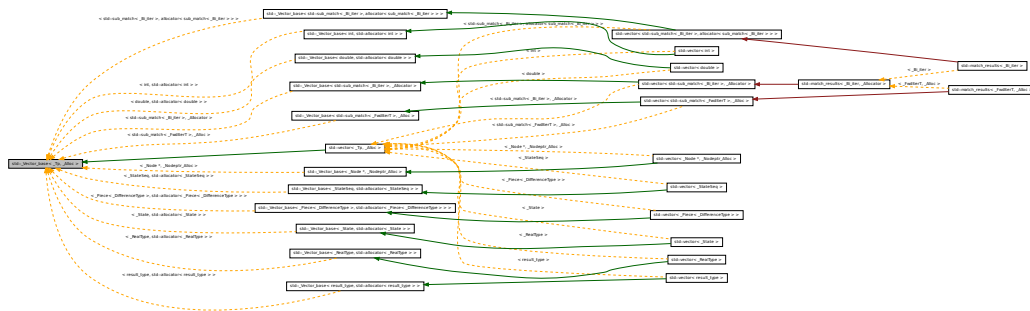
The documentation for this struct was generated from the following file:

- [tuple](#)

5.320 std::_Vector_base< _Tp, _Alloc > Struct Template Reference

See [bits/stl_deque.h](#)'s `_Deque_base` for an explanation.

Inheritance diagram for `std::_Vector_base< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc::template rebind< _Tp >::other _Tp_alloc_type`
- typedef `_Alloc allocator_type`

Public Member Functions

- `_Vector_base` (const `allocator_type` &__a)
- `_Vector_base` (`_Vector_base` &&__x)
- `_Vector_base` (size_t __n)

- **_Vector_base** (size_t __n, const allocator_type &__a)
- **_Tp_alloc_type::pointer _M_allocate** (size_t __n)
- **void _M_deallocate** (typename _Tp_alloc_type::pointer __p, size_t __n)
- **_Tp_alloc_type & _M_get_Tp_allocator** ()
- **const _Tp_alloc_type & _M_get_Tp_allocator** () const
- **allocator_type get_allocator** () const

Public Attributes

- **_Vector_impl _M_impl**

5.320.1 Detailed Description

template<typename _Tp, typename _Alloc> struct std::_Vector_base< _Tp, _Alloc >

See [bits/stl_deque.h](#)'s [_Deque_base](#) for an explanation.

Definition at line 69 of file [stl_vector.h](#).

The documentation for this struct was generated from the following file:

- [stl_vector.h](#)

5.321 std::_Weak_result_type< _Functor > Struct Template Reference

Inheritance diagram for `std::_Weak_result_type< _Functor >`:



5.321.1 Detailed Description

template<typename _Functor> struct std::_Weak_result_type< _Functor >

Strip top-level cv-qualifiers from the function object and let [_Weak_result_type_impl](#) perform the real work.

Definition at line 168 of file [functional](#).

The documentation for this struct was generated from the following file:

5.322 `std::_Weak_result_type_impl<_Functor>` Struct Template Reference 469

- [functional](#)

5.322 `std::_Weak_result_type_impl<_Functor>` Struct Template Reference

Inheritance diagram for `std::_Weak_result_type_impl<_Functor>`:



5.322.1 Detailed Description

`template<typename _Functor> struct std::_Weak_result_type_impl<_Functor>`

Base class for any function object that has a weak result type, as defined in 3.3/3 of TR1.

Definition at line 110 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.323 `std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function reference.

Public Types

- `typedef _Res result_type`

5.323.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>`

Retrieve the result type for a function reference.

Definition at line 123 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.324 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function pointer.

Public Types

- `typedef _Res result_type`

5.324.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>
```

Retrieve the result type for a function pointer.

Definition at line 130 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.325 `std::_Weak_result_type_impl< _Res(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function type.

Public Types

- `typedef _Res result_type`

5.326 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >`
Struct Template Reference 1471

5.325.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(_ArgTypes...)>`

Retrieve the result type for a function type.

Definition at line 116 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.326 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >` **Struct Template Reference**

Retrieve result type for a const member function pointer.

Public Types

- `typedef _Res result_type`

5.326.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >`

Retrieve result type for a const member function pointer.

Definition at line 144 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.327 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >` **Struct Template Reference**

Retrieve result type for a const volatile member function pointer.

Public Types

- `typedef _Res result_type`

5.327.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >
```

Retrieve result type for a const volatile member function pointer.

Definition at line 158 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.328 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile > Struct Template Reference`

Retrieve result type for a volatile member function pointer.

Public Types

- `typedef _Res result_type`

5.328.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >
```

Retrieve result type for a volatile member function pointer.

Definition at line 151 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.329 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)> Struct`
Template Reference 1473

5.329 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)> Struct` Template Reference

Retrieve result type for a member function pointer.

Public Types

- `typedef _Res result_type`

5.329.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct  
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)>
```

Retrieve result type for a member function pointer.

Definition at line 137 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.330 `std::add_lvalue_reference< _Tp > Struct` Template Reference

[add_lvalue_reference](#)

Inherits `std::__add_lvalue_reference_helper< _Tp >`.

Public Types

- `typedef _Tp type`

5.330.1 Detailed Description

```
template<typename _Tp> struct std::add_lvalue_reference< _Tp >
```

[add_lvalue_reference](#)

Definition at line 125 of file type_traits.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.331 `std::add_rvalue_reference< _Tp >` Struct Template Reference

[add_rvalue_reference](#)

Inherits `std::__add_rvalue_reference_helper< _Tp >`.

Public Types

- `typedef _Tp type`

5.331.1 Detailed Description

`template<typename _Tp> struct std::add_rvalue_reference< _Tp >`

[add_rvalue_reference](#)

Definition at line 140 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.332 `std::adopt_lock_t` Struct Reference

Assume the calling thread has already obtained mutex ownership /// and manage it.

5.332.1 Detailed Description

Assume the calling thread has already obtained mutex ownership /// and manage it.

Definition at line 382 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

5.333 `std::aligned_storage< _Len, _Align >` Struct Template Reference

Alignment type.

5.333.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = __alignof__(typename __-
aligned_storage_msa<_Len>::__type)> struct std::aligned_storage< _Len, _-
Align >
```

Alignment type. The value of `_Align` is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than `_Len` (3.9). The member typedef type shall be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

Definition at line 370 of file `type_traits`.

The documentation for this struct was generated from the following file:

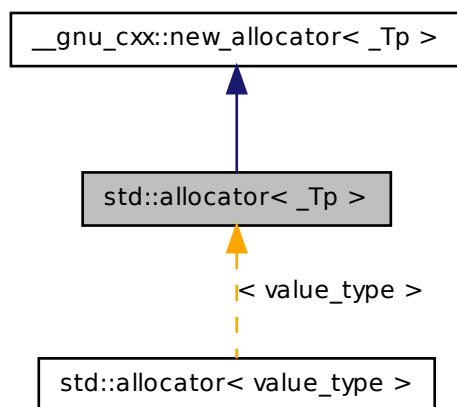
- [type_traits](#)

5.334 `std::allocator< _Tp >` Class Template Reference

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Inheritance diagram for `std::allocator< _Tp >`:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **allocator** (const [allocator](#) &__a) throw ()
- template<typename _Tp1 >
 allocator (const [allocator](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, const void * = 0)
- void **construct** (pointer __p, const _Tp & __val)

- `template<typename... _Args>`
`void construct (pointer __p, _Args &&...__args)`
- `void deallocate (pointer __p, size_type)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.334.1 Detailed Description

`template<typename _Tp> class std::allocator< _Tp >`

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Definition at line 86 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

5.335 `std::allocator< void >` Class Template Reference

[allocator<void>](#) specialization.

Public Types

- `typedef const void * const_pointer`
- `typedef ptrdiff_t difference_type`
- `typedef void * pointer`
- `typedef size_t size_type`
- `typedef void value_type`

5.335.1 Detailed Description

`template<> class std::allocator< void >`

[allocator<void>](#) specialization.

Definition at line 64 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

5.336 `std::atomic<_Tp>` Struct Template Reference

`atomic` /// 29.4.3, Generic atomic type, primary class template.

Public Member Functions

- `atomic` (`_Tp __i`)
- `atomic` (`const atomic &`)
- `bool compare_exchange_strong` (`_Tp &, _Tp, memory_order, memory_order`) volatile
- `bool compare_exchange_strong` (`_Tp &, _Tp, memory_order=memory_order_seq_cst`) volatile
- `bool compare_exchange_weak` (`_Tp &, _Tp, memory_order=memory_order_seq_cst`) volatile
- `bool compare_exchange_weak` (`_Tp &, _Tp, memory_order, memory_order`) volatile
- `_Tp exchange` (`_Tp __i, memory_order=memory_order_seq_cst`) volatile
- `bool is_lock_free` () const volatile
- `_Tp load` (`memory_order=memory_order_seq_cst`) const volatile
- `operator _Tp` () const
- `atomic & operator=` (`const atomic &`) volatile
- `_Tp operator=` (`_Tp __i`)
- `void store` (`_Tp, memory_order=memory_order_seq_cst`) volatile

5.336.1 Detailed Description

`template<typename _Tp> struct std::atomic<_Tp>`

`atomic` /// 29.4.3, Generic atomic type, primary class template.

Definition at line 86 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.337 `std::atomic<_Tp*>` Struct Template Reference

Partial specialization for pointer types.

Inherits `atomic_address`.

Public Member Functions

- **atomic** (_Tp *__v)
- **atomic** (const [atomic](#) &)
- bool **compare_exchange_strong** (_Tp * &, _Tp *, [memory_order](#), [memory_order](#))
- bool **compare_exchange_strong** (_Tp * &, _Tp *, [memory_order](#)=memory_order_seq_cst)
- bool **compare_exchange_weak** (_Tp * &, _Tp *, [memory_order](#), [memory_order](#))
- bool **compare_exchange_weak** (_Tp * &, _Tp *, [memory_order](#)=memory_order_seq_cst)
- _Tp * **exchange** (_Tp *, [memory_order](#)=memory_order_seq_cst)
- _Tp * **fetch_add** (ptrdiff_t, [memory_order](#)=memory_order_seq_cst)
- _Tp * **fetch_sub** (ptrdiff_t, [memory_order](#)=memory_order_seq_cst)
- _Tp * **load** ([memory_order](#)=memory_order_seq_cst) const
- **operator _Tp** * () const
- _Tp * **operator++** (int)
- _Tp * **operator++** ()
- _Tp * **operator+=** (ptrdiff_t __d)
- _Tp * **operator--** ()
- _Tp * **operator--** (int)
- _Tp * **operator-=** (ptrdiff_t __d)
- _Tp * **operator=** (_Tp *__v)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- void **store** (_Tp *, [memory_order](#)=memory_order_seq_cst)

5.337.1 Detailed Description

template<typename _Tp> struct std::atomic< _Tp * >

Partial specialization for pointer types.

Definition at line 134 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.338 std::atomic< bool > Struct Template Reference

Explicit specialization for bool.

Inherits `__atomic_bool_base`.

Public Types

- typedef atomic_bool __base_type
- typedef bool __integral_type

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- bool _M_i

5.338.1 Detailed Description

template<> struct std::atomic< bool >

Explicit specialization for bool.

Definition at line 222 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.339 std::atomic< char > Struct Template Reference

Explicit specialization for char.

Inherits __atomic_char_base.

Public Types

- typedef [atomic_char](#) __base_type
- typedef char __integral_type

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- `char _M_i`

5.339.1 Detailed Description

`template<> struct std::atomic< char >`

Explicit specialization for `char`.

Definition at line 240 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.340 `std::atomic< char16_t >` Struct Template Reference

Explicit specialization for `char16_t`.

Inherits `__atomic_short_base`.

Public Types

- typedef `atomic_char16_t __base_type`
- typedef `char16_t __integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- `short _M_i`

5.340.1 Detailed Description

template<> struct std::atomic< char16_t >

Explicit specialization for char16_t.

Definition at line 456 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.341 std::atomic< char32_t > Struct Template Reference

Explicit specialization for char32_t.

Inherits `__atomic_int_base`.

Public Types

- typedef [atomic_char32_t](#) `__base_type`
- typedef char32_t `__integral_type`

Public Member Functions

- **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- `int _M_i`

5.341.1 Detailed Description

`template<> struct std::atomic< char32_t >`

Explicit specialization for `char32_t`.

Definition at line 474 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.342 `std::atomic< int >` Struct Template Reference

Explicit specialization for `int`.

Inherits `__atomic_int_base`.

Public Types

- typedef [atomic_int](#) `__base_type`
- typedef `int` `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

Public Attributes

- `int _M_i`

5.342.1 Detailed Description

`template<> struct std::atomic< int >`

Explicit specialization for `int`.

Definition at line 330 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.343 `std::atomic< long >` Struct Template Reference

Explicit specialization for long.

Inherits `__atomic_long_base`.

Public Types

- typedef `atomic_long` `__base_type`
- typedef long `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- long `_M_i`

5.343.1 Detailed Description

`template<> struct std::atomic< long >`

Explicit specialization for long.

Definition at line 366 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.344 `std::atomic< long long >` Struct Template Reference

Explicit specialization for long long.

Inherits `__atomic_llong_base`.

Public Types

- typedef [atomic_llong](#) __base_type
- typedef long long __integral_type

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- long long _M_i

5.344.1 Detailed Description

template<> struct std::atomic< long long >

Explicit specialization for long long.

Definition at line 402 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.345 std::atomic< short > Struct Template Reference

Explicit specialization for short.

Inherits __atomic_short_base.

Public Types

- typedef [atomic_short](#) __base_type
- typedef short __integral_type

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- short **_M_i**

5.345.1 Detailed Description

template<> struct std::atomic< short >

Explicit specialization for short.

Definition at line 294 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.346 **std::atomic< signed char > Struct Template Reference**

Explicit specialization for signed char.

Inherits `__atomic_schar_base`.

Public Types

- typedef [atomic_schar](#) **__base_type**
- typedef signed char **__integral_type**

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- signed char `_M_i`

5.346.1 Detailed Description

`template<> struct std::atomic< signed char >`

Explicit specialization for signed char.

Definition at line 258 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.347 `std::atomic< unsigned char >` Struct Template Reference

Explicit specialization for unsigned char.

Inherits `__atomic_uchar_base`.

Public Types

- typedef [atomic_uchar](#) `__base_type`
- typedef unsigned char `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile

Public Attributes

- unsigned char `_M_i`

5.347.1 Detailed Description

template<> struct std::atomic< unsigned char >

Explicit specialization for unsigned char.

Definition at line 276 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.348 std::atomic< unsigned int > Struct Template Reference

Explicit specialization for unsigned int.

Inherits `__atomic_uint_base`.

Public Types

- typedef [atomic_uint](#) `__base_type`
- typedef unsigned int `__integral_type`

Public Member Functions

- **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- unsigned int `_M_i`

5.348.1 Detailed Description

template<> struct std::atomic< unsigned int >

Explicit specialization for unsigned int.

Definition at line 348 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.349 `std::atomic< unsigned long >` Struct Template Reference

Explicit specialization for unsigned long.

Inherits `__atomic_ulong_base`.

Public Types

- typedef [atomic_ulong](#) `__base_type`
- typedef unsigned long `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile

Public Attributes

- unsigned long `_M_i`

5.349.1 Detailed Description

`template<> struct std::atomic< unsigned long >`

Explicit specialization for unsigned long.

Definition at line 384 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.350 `std::atomic< unsigned long long >` Struct Template Reference

Explicit specialization for unsigned long long.

Inherits `__atomic_ullong_base`.

Public Types

- typedef `atomic_ullong` `__base_type`
- typedef unsigned long long `__integral_type`

Public Member Functions

- `atomic` (`__integral_type` `__i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- unsigned long long `_M_i`

5.350.1 Detailed Description

`template<> struct std::atomic< unsigned long long >`

Explicit specialization for unsigned long long.

Definition at line 420 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.351 `std::atomic< unsigned short >` Struct Template Reference

Explicit specialization for unsigned short.

Inherits `__atomic_ushort_base`.

Public Types

- typedef `atomic_ushort` `__base_type`
- typedef unsigned short `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- unsigned short `_M_i`

5.351.1 Detailed Description

`template<> struct std::atomic< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 312 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.352 `std::atomic< void * >` Struct Template Reference

Explicit specialization for `void*`.

Inherits `atomic_address`.

Public Types

- `typedef atomic_address __base_type`
- `typedef void * __integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

5.352.1 Detailed Description

template<> struct std::atomic< void * >

Explicit specialization for void*.

Definition at line 204 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.353 std::atomic< wchar_t > Struct Template Reference

Explicit specialization for wchar_t.

Inherits `__atomic_wchar_t_base`.

Public Types

- typedef [atomic_wchar_t](#) `__base_type`
- typedef wchar_t `__integral_type`

Public Member Functions

- **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- wchar_t `_M_i`

5.353.1 Detailed Description

template<> struct std::atomic< wchar_t >

Explicit specialization for wchar_t.

Definition at line 438 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.354 `std::auto_ptr< _Tp >` Class Template Reference

A simple smart pointer providing strict ownership semantics.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `auto_ptr (element_type *__p=0) throw ()`
- `auto_ptr (auto_ptr &__a) throw ()`
- `template<typename _Tp1 >
 auto_ptr (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr (auto_ptr_ref< element_type > __ref) throw ()`
- `~auto_ptr ()`
- `element_type * get () const throw ()`
- `template<typename _Tp1 >
 operator auto_ptr< _Tp1 > () throw ()`
- `template<typename _Tp1 >
 operator auto_ptr_ref< _Tp1 > () throw ()`
- `element_type & operator* () const throw ()`
- `element_type * operator-> () const throw ()`
- `auto_ptr & operator= (auto_ptr &__a) throw ()`
- `template<typename _Tp1 >
 auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr & operator= (auto_ptr_ref< element_type > __ref) throw ()`
- `element_type * release () throw ()`
- `void reset (element_type *__p=0) throw ()`

5.354.1 Detailed Description

`template<typename _Tp> class std::auto_ptr< _Tp >`

A simple smart pointer providing strict ownership semantics. The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` 127. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 85 of file `auto_ptr.h`.

5.354.2 Member Typedef Documentation

5.354.2.1 `template<typename _Tp> typedef _Tp std::auto_ptr< _Tp >::element_type`

The pointed-to type.

Definition at line 92 of file `auto_ptr.h`.

5.354.3 Constructor & Destructor Documentation

5.354.3.1 `template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr (element_type * __p = 0) throw () [inline, explicit]`

An `auto_ptr` is usually constructed from a raw pointer.

Parameters

p A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*.

Definition at line 101 of file `auto_ptr.h`.

5.354.3.2 `template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr (auto_ptr< _Tp > & __a) throw () [inline]`

An `auto_ptr` can be constructed from another `auto_ptr`.

Parameters

a Another auto_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 110 of file auto_ptr.h.

5.354.3.3 `template<typename _Tp> template<typename _Tp1 >
std::auto_ptr< _Tp >::auto_ptr (auto_ptr< _Tp1 > & __a) throw
() [inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

a Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 123 of file auto_ptr.h.

5.354.3.4 `template<typename _Tp> std::auto_ptr< _Tp >::~~auto_ptr ()
[inline]`

When the auto_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 168 of file auto_ptr.h.

5.354.3.5 `template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr (
auto_ptr_ref< element_type > __ref) throw () [inline]`

Automatic conversions.

These operations convert an auto_ptr into and from an `auto_ptr_ref` automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(.....);  
...  
auto_ptr<Base> ptr = func_returning_auto_ptr(.....);
```

Definition at line 258 of file auto_ptr.h.

5.354.4 Member Function Documentation

5.354.4.1 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::get
(void) const throw () [inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This `auto_ptr` still owns the memory.

Definition at line 209 of file `auto_ptr.h`.

5.354.4.2 `template<typename _Tp> element_type& std::auto_ptr<_Tp>
>::operator* () const throw () [inline]`

Smart pointer dereferencing.

If this `auto_ptr` no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 179 of file `auto_ptr.h`.

5.354.4.3 `template<typename _Tp> element_type* std::auto_ptr<_Tp>
>::operator-> () const throw () [inline]`

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 192 of file `auto_ptr.h`.

5.354.4.4 `template<typename _Tp> template<typename _Tp1> auto_ptr&
std::auto_ptr<_Tp>::operator= (auto_ptr<_Tp1> & __a)
throw () [inline]`

`auto_ptr` assignment operator.

Parameters

a Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 152 of file auto_ptr.h.

5.354.4.5 `template<typename _Tp> auto_ptr& std::auto_ptr< _Tp
>::operator= (auto_ptr< _Tp > & __a) throw () [inline]`

auto_ptr assignment operator.

Parameters

a Another auto_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 134 of file auto_ptr.h.

5.354.4.6 `template<typename _Tp> element_type* std::auto_ptr< _Tp
>::release () throw () [inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 223 of file auto_ptr.h.

5.354.4.7 `template<typename _Tp> void std::auto_ptr< _Tp >::reset (element_type * __p = 0) throw () [inline]`

Forcibly deletes the managed object.

Parameters

p A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*. The previous object has been deleted.

Definition at line 238 of file `auto_ptr.h`.

The documentation for this class was generated from the following file:

- [auto_ptr.h](#)

5.355 `std::auto_ptr_ref< _Tp1 >` Struct Template Reference

Public Member Functions

- `auto_ptr_ref(_Tp1 * __p)`

Public Attributes

- `_Tp1 * _M_ptr`

5.355.1 Detailed Description

`template<typename _Tp1> struct std::auto_ptr_ref< _Tp1 >`

A wrapper class to provide [auto_ptr](#) with reference semantics. For example, an [auto_ptr](#) can be assigned (or constructed from) the result of a function which returns an [auto_ptr](#) by value.

All the [auto_ptr_ref](#) stuff should happen behind the scenes.

Definition at line 46 of file `auto_ptr.h`.

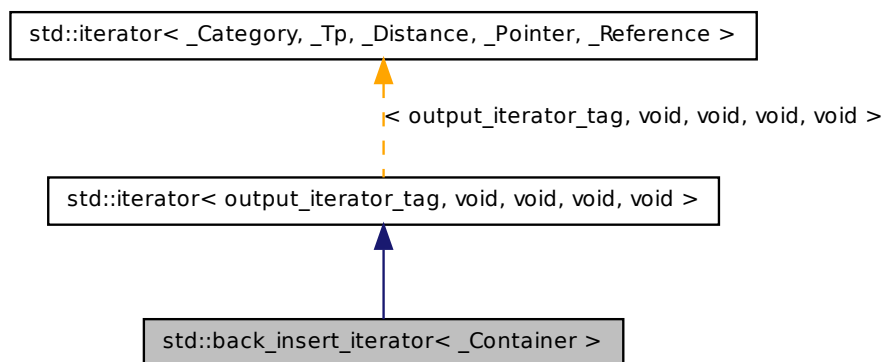
The documentation for this struct was generated from the following file:

- [auto_ptr.h](#)

5.356 `std::back_insert_iterator<_Container>` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::back_insert_iterator<_Container>`:



Public Types

- `typedef _Container container_type`
- `typedef void difference_type`
- `typedef output_iterator_tag iterator_category`
- `typedef void pointer`
- `typedef void reference`
- `typedef void value_type`

Public Member Functions

- `back_insert_iterator(_Container &__x)`
- `back_insert_iterator & operator* ()`
- `back_insert_iterator & operator++ ()`
- `back_insert_iterator operator++ (int)`
- `back_insert_iterator & operator= (typename _Container::const_reference __value)`

- [back_insert_iterator](#) & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**

5.356.1 Detailed Description

template<typename _Container> class std::back_insert_iterator< _Container >

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using push_back.

Tip: Using the back_inserter function to create these iterators can save typing.

Definition at line 394 of file stl_iterator.h.

5.356.2 Member Typedef Documentation

5.356.2.1 **template<typename _Container > typedef _Container
std::back_insert_iterator< _Container >::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 402 of file stl_iterator.h.

5.356.2.2 **typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::difference_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

5.356.2.3 **typedef output_iterator_tag std::iterator< output_iterator_tag , void
, void , void , void >::iterator_category [inherited]**

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

5.356.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 116 of file stl_iterator_base_types.h.

5.356.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.356.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file stl_iterator_base_types.h.

5.356.3 Constructor & Destructor Documentation

5.356.3.1 `template<typename _Container > std::back_insert_iterator< _Container >::back_insert_iterator (_Container & __x)
[inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 406 of file stl_iterator.h.

5.356.4 Member Function Documentation

5.356.4.1 `template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator* ()
[inline]`

Simply returns *this.

Definition at line 437 of file stl_iterator.h.

5.356.4.2 `template<typename _Container > back_insert_iterator
std::back_insert_iterator< _Container >::operator++ (int)
[inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 447 of file stl_iterator.h.

5.356.4.3 `template<typename _Container > back_insert_iterator&
std::back_insert_iterator< _Container >::operator++ ()
[inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 442 of file stl_iterator.h.

5.356.4.4 `template<typename _Container > back_insert_iterator&
std::back_insert_iterator< _Container >::operator= (typename
_Container::const_reference __value) [inline]`

Parameters

value An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

Definition at line 420 of file stl_iterator.h.

The documentation for this class was generated from the following file:

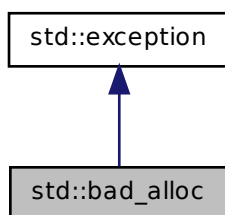
- [stl_iterator.h](#)

5.357 std::bad_alloc Class Reference

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Inheritance diagram for std::bad_alloc:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.357.1 Detailed Description

Exception possibly thrown by `new`.

[bad_alloc](#) (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 56 of file `new`.

5.357.2 Member Function Documentation

5.357.2.1 virtual const char* std::bad_alloc::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

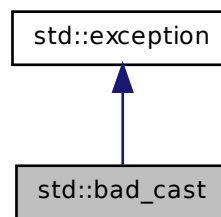
- [new](#)

5.358 `std::bad_cast` Class Reference

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Inheritance diagram for `std::bad_cast`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.358.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 174 of file `typeinfo`.

5.358.2 Member Function Documentation

5.358.2.1 virtual const char* `std::bad_cast::what` () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

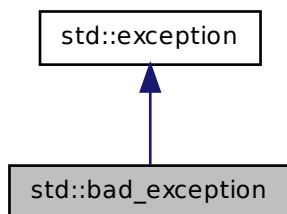
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

5.359 `std::bad_exception` Class Reference

Inheritance diagram for `std::bad_exception`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.359.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 74 of file `exception`.

5.359.2 Member Function Documentation

5.359.2.1 virtual const char* `std::bad_exception::what` () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

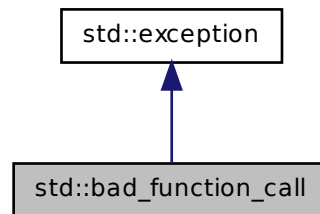
The documentation for this class was generated from the following file:

- [exception](#)

5.360 std::bad_function_call Class Reference

Exception class thrown when class template function's operator() is called with an empty target.

Inheritance diagram for std::bad_function_call:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.360.1 Detailed Description

Exception class thrown when class template function's operator() is called with an empty target.

Definition at line 1384 of file functional.

5.360.2 Member Function Documentation

5.360.2.1 virtual const char* std::exception::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and [std::ios_base::failure](#).

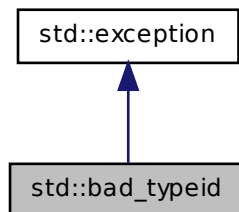
The documentation for this class was generated from the following file:

- [functional](#)

5.361 `std::bad_typeid` Class Reference

Thrown when a NULL pointer in a `typeid` expression is used.

Inheritance diagram for `std::bad_typeid`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.361.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 191 of file `typeinfo`.

Public Member Functions

- [basic_filebuf](#) ()
- virtual [~basic_filebuf](#) ()
- [__filebuf_type](#) * [close](#) ()
- [streamsize](#) [in_avail](#) ()
- bool [is_open](#) () const throw ()
- [__filebuf_type](#) * [open](#) (const char * __s, [ios_base::openmode](#) __mode)
- [__filebuf_type](#) * [open](#) (const [std::string](#) & __s, [ios_base::openmode](#) __mode)
- [int_type](#) [sbumpc](#) ()
- [int_type](#) [sgetc](#) ()
- [streamsize](#) [sgetn](#) ([char_type](#) * __s, [streamsize](#) __n)
- [int_type](#) [snextc](#) ()
- [int_type](#) [sputbackc](#) ([char_type](#) __c)
- [int_type](#) [sputc](#) ([char_type](#) __c)
- [streamsize](#) [sputn](#) (const [char_type](#) * __s, [streamsize](#) __n)
- void [stossc](#) ()
- [int_type](#) [sungetc](#) ()

Protected Member Functions

- void [_M_allocate_internal_buffer](#) ()
- bool [_M_convert_to_external](#) ([char_type](#) *, [streamsize](#))
- void [_M_create_pback](#) ()
- void [_M_destroy_internal_buffer](#) () throw ()
- void [_M_destroy_pback](#) () throw ()
- [pos_type](#) [_M_seek](#) ([off_type](#) __off, [ios_base::seekdir](#) __way, [__state_type](#) __state)
- void [_M_set_buffer](#) ([streamsize](#) __off)
- bool [_M_terminate_output](#) ()
- void [gbump](#) (int __n)
- virtual void [imbue](#) (const [locale](#) & __loc)
- virtual [int_type](#) [overflow](#) ([int_type](#) __c= [_Traits::eof](#)())
- virtual [int_type](#) [overflow](#) ([int_type](#)=[traits_type::eof](#)())
- virtual [int_type](#) [pbackfail](#) ([int_type](#) __c= [_Traits::eof](#)())
- virtual [int_type](#) [pbackfail](#) ([int_type](#)=[traits_type::eof](#)())
- void [pbump](#) (int __n)
- virtual [pos_type](#) [seekoff](#) ([off_type](#) __off, [ios_base::seekdir](#) __way, [ios_base::openmode](#) __mode=[ios_base::in](#)|[ios_base::out](#))
- virtual [pos_type](#) [seekoff](#) ([off_type](#), [ios_base::seekdir](#), [ios_base::openmode](#)=[ios_base::in](#)|[ios_base::out](#))
- virtual [pos_type](#) [seekpos](#) ([pos_type](#), [ios_base::openmode](#)=[ios_base::in](#)|[ios_base::out](#))

- virtual `pos_type seekpos` (`pos_type __pos`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- virtual `__streambuf_type * setbuf` (`char_type *__s`, `streamsize __n`)
- virtual `basic_streambuf< char_type, _Traits > * setbuf` (`char_type *`, `streamsize`)
- void `setg` (`char_type *__gbeg`, `char_type *__gnext`, `char_type *__gend`)
- void `setp` (`char_type *__pbeg`, `char_type *__pend`)
- virtual `streamsize showmanyc` ()
- virtual int `sync` ()
- virtual `int_type uflow` ()
- virtual `int_type underflow` ()
- virtual `streamsize xsgetn` (`char_type *__s`, `streamsize __n`)
- virtual `streamsize xsgetn` (`char_type *__s`, `streamsize __n`)
- virtual `streamsize xsputn` (`const char_type *__s`, `streamsize __n`)
- virtual `streamsize xspn` (`const char_type *__s`, `streamsize __n`)
- `char_type * eback` () const
- `char_type * gptr` () const
- `char_type * egptr` () const
- `char_type * pbase` () const
- `char_type * pptr` () const
- `char_type * epptr` () const

Protected Attributes

- `char_type * _M_buf`
- bool `_M_buf_allocated`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- bool `_M_reading`
- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- bool `_M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- bool `_M_pback_init`

Friends

- template<bool _IsMove, typename _CharT2 >
[__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-](#)
[type __copy_move_a2 \(istreambuf_iterator< _CharT2 >, istreambuf_iterator<](#)
[_CharT2 >, _CharT2 *\)](#)
 - [streamsize __copy_streambufs_eof \(__streambuf_type *, __streambuf_type *,](#)
[bool &\)](#)
 - class **basic_ios**< **char_type**, **traits_type** >
 - class **basic_istream**< **char_type**, **traits_type** >
 - class **basic_ostream**< **char_type**, **traits_type** >
 - template<typename _CharT2 >
[__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_-](#)
[iterator< _CharT2 > >::__type find \(istreambuf_iterator< _CharT2 >, -](#)
[istreambuf_iterator< _CharT2 >, const _CharT2 &\)](#)
 - template<typename _CharT2, typename _Traits2, typename _Alloc >
[basic_istream< _CharT2, _Traits2 > & getline \(basic_istream< _CharT2, _-](#)
[Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2\)](#)
 - class **ios_base**
 - class **istreambuf_iterator**< **char_type**, **traits_type** >
 - template<typename _CharT2, typename _Traits2 >
[basic_istream< _CharT2, _Traits2 > & operator>> \(basic_istream< _CharT2,](#)
[_Traits2 > &, _CharT2 *\)](#)
 - template<typename _CharT2, typename _Traits2, typename _Alloc >
[basic_istream< _CharT2, _Traits2 > & operator>> \(basic_istream< _CharT2,](#)
[_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &\)](#)
 - class **ostreambuf_iterator**< **char_type**, **traits_type** >
-
- [char_type * _M_in_beg](#)
 - [char_type * _M_in_cur](#)
 - [char_type * _M_in_end](#)
 - [char_type * _M_out_beg](#)
 - [char_type * _M_out_cur](#)
 - [char_type * _M_out_end](#)
 - [locale _M_buf_locale](#)
 - [locale pubimbue \(const locale &__loc\)](#)
 - [locale getloc \(\) const](#)
 - [__streambuf_type * pubsetbuf \(char_type *__s, streamsize __n\)](#)
 - [pos_type pubseekoff \(off_type __off, ios_base::seekdir __way, ios_-](#)
[base::openmode __mode=ios_base::in|ios_base::out\)](#)
 - [pos_type pubseekpos \(pos_type __sp, ios_base::openmode __mode=ios_-](#)
[base::in|ios_base::out\)](#)
 - [int pubsync \(\)](#)

5.362.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_filebuf< _  
CharT, _Traits >
```

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Definition at line 67 of file `fstream`.

5.362.2 Member Typedef Documentation

```
5.362.2.1 template<typename _CharT, typename _Traits> typedef  
basic_streambuf<char_type, traits_type> std::basic_filebuf<  
_CharT, _Traits >::__streambuf_type
```

This is a non-standard type.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 77 of file `fstream`.

```
5.362.2.2 template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_filebuf< _CharT, _Traits >::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Reimplemented in [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#).

Definition at line 71 of file `fstream`.

```
5.362.2.3 template<typename _CharT, typename _Traits> typedef  
traits_type::int_type std::basic_filebuf< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Reimplemented in [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#).

Definition at line 73 of file `fstream`.

5.362.2.4 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_filebuf<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 75 of file `fstream`.

5.362.2.5 `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_filebuf<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::enc_filebuf<_CharT>](#), and [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 74 of file `fstream`.

5.362.2.6 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_filebuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::enc_filebuf<_CharT>](#), and [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 72 of file `fstream`.

5.362.3 Constructor & Destructor Documentation

5.362.3.1 `template<typename _CharT, typename _Traits> std::basic_filebuf<_CharT, _Traits>::basic_filebuf ()`

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 79 of file fstream.tcc.

References `std::basic_streambuf< _CharT, _Traits >::_M_buf_locale`.

5.362.3.2 `template<typename _CharT, typename _Traits> virtual std::basic_filebuf< _CharT, _Traits >::~~basic_filebuf () [inline, virtual]`

The destructor closes the file first.

Definition at line 214 of file fstream.

5.362.4 Member Function Documentation

5.362.4.1 `template<typename _CharT, typename _Traits> void std::basic_filebuf< _CharT, _Traits >::_M_create_pback () [inline, protected]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file fstream.

5.362.4.2 `template<typename _CharT, typename _Traits> void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw () [inline, protected]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.4.3 `template<typename _CharT, typename _Traits> void
std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (streamsize
__off) [inline, protected]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

__off == `egptr()` - `eback()` upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `eptr()` - `pbase()` == `_M_buf_size` - 1, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 387 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.4.4 `template<typename _CharT, typename _Traits> basic_filebuf<
_CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT,
_Traits >::close ()`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::is_open()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.362.4.5 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eback () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.362.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::egptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.362.4.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::epptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.362.4.8 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.362.4.9 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.362.4.10 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.4.11 `template<typename _CharT, typename _Traits> void
std::basic_filebuf< _CharT, _Traits >::imbue (const locale &)
[protected, virtual]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 855 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.362.4.12 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::in_avail ()
[inline, inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file streambuf.

5.362.4.13 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::is_open () const throw ()
[inline]`

Returns true if the external file is open.

Definition at line 222 of file fstream.

Referenced by std::basic_filebuf< _CharT, _Traits >::close(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::setbuf(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), and __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf().

5.362.4.14 `template<typename _CharT, typename _Traits > basic_filebuf<
_CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT,
_Traits >::open (const char * __s, ios_base::openmode __mode)`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Returns

this on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +-----

	ios_base	Flag combination	stdio equivalent		binary	in	out	trunc	app
+-----+		w		+	w		+	+	a
+-----+		a		+	a		+	+	w
+-----+		r		+	r		+	+	r
+-----+		w		+	w		+	+	a
+-----+		a		+	a		+	+	w
+-----+		r		+	r		+	+	r
+-----+		w		+	w		+	+	a
+-----+		a		+	a		+	+	w
+-----+		r		+	r		+	+	r
+-----+		w		+	w		+	+	a
+-----+		a		+	a		+	+	w
+-----+		r		+	r		+	+	r
+-----+		w		+	w		+	+	a
+-----+		a		+	a		+	+	w

Definition at line 94 of file fstream.tcc.

References std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), std::ios_base::ate, std::basic_filebuf< _CharT, _Traits >::close(), std::ios_base::end, and std::basic_filebuf< _CharT, _Traits >::is_open().

5.362.4.15 `template<typename _CharT, typename _Traits> __filebuf_type*
std::basic_filebuf< _CharT, _Traits >::open (const std::string &
__s, ios_base::openmode __mode) [inline]`

Opens an external file.

Parameters

- s* The name of the file.
- mode* The open mode flags.

Returns

`this` on success, NULL on failure

Definition at line 275 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

5.362.4.16 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::overflow (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

- c* An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#).

Definition at line 746 of file streambuf.

Referenced by [std::basic_streambuf<_CharT, _Traits>::xsputn\(\)](#).

5.362.4.17 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

Note

Base class version does nothing, returns eof().

Reimplemented in [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#).

Definition at line 702 of file streambuf.

5.362.4.18 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

5.362.4.19 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

n The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.362.4.20 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.362.4.21 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 205 of file streambuf.

5.362.4.22 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 239 of file streambuf.

5.362.4.23 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 244 of file streambuf.

5.362.4.24 `template<typename _CharT, typename _Traits>
 __streambuf_type* std::basic_streambuf< _CharT, _Traits
 >::pubsetbuf(char_type * __s, streamsize __n) [inline,
 inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.362.4.25 `template<typename _CharT, typename _Traits> int
 std::basic_streambuf< _CharT, _Traits >::pubsync ()
 [inline, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

5.362.4.26 `template<typename _CharT, typename _Traits> int_type
 std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline,
 inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

5.362.4.27 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.362.4.28 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos(pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.362.4.29 `template<typename _CharT, typename _Traits> basic_filebuf<
_CharT, _Traits >::__streambuf_type * std::basic_filebuf<
_CharT, _Traits >::setbuf(char_type * __s, streamsize __n)
[protected, virtual]`

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 657 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.362.4.30 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT,_Traits>::setbuf(char_type *, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

5.362.4.31 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setg(char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

5.362.4.32 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setp (char_type
* __pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.362.4.33 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sgetc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.362.4.34 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sgetn (char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.362.4.35 `template<typename _CharT, typename _Traits> streamsize
std::basic_filebuf< _CharT, _Traits >::showmanyc ()
[protected, virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 178 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::M_mode`, `std::ios_base::binary`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.362.4.36 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::snextc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.362.4.37 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

`c` The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but `c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `c`.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

5.362.4.38 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::putc (char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

Parameters

`c` A character to output.

Returns

`c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `c` in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

5.362.4.39 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sputn (const char_type
* __s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file streambuf.

5.362.4.40 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::stoss () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file streambuf.

5.362.4.41 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

5.362.4.42 `template<typename _CharT , typename _Traits > int
std::basic_filebuf< _CharT, _Traits >::sync () [protected,
virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 838 of file fstream.tcc.

References [std::basic_streambuf< _CharT, _Traits >::pbase\(\)](#), and [std::basic_streambuf< _CharT, _Traits >::pptr\(\)](#).

5.362.4.43 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 678 of file streambuf.

5.362.4.44 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::underflow() [protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 204 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in()`, `std::ios_base::in`, and `std::min()`.

5.362.4.45 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::xsgetn(char_type * __s, streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

5.362.4.46 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::xsputn (const
char_type * __s, streamsize __n) [protected, virtual,
inherited]`

Multiple character insertion.

Parameters

s A buffer area.

n Maximum number of characters to write.

Returns

The number of characters written.

Writes `s[0]` through `s[n-1]` to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.362.5 Member Data Documentation

5.362.5.1 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_buf [protected]`

Pointer to the beginning of internal buffer.

Definition at line 109 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::setbuf()`.

5.362.5.2 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::_M_buf_locale
[protected, inherited]`

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`.

5.362.5.3 `template<typename _CharT, typename _Traits> size_t
std::basic_filebuf< _CharT, _Traits >::_M_buf_size [protected]`

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::setbuf()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.5.4 `template<typename _CharT, typename _Traits> char*
std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]`

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to [eback\(\)](#).

Definition at line 151 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.5.5 `template<typename _CharT, typename _Traits> streamsize
std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size
[protected]`

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.5.6 `template<typename _CharT, typename _Traits> const
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next
[protected]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to [egptr\(\)](#).

Definition at line 163 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_beg
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.362.5.8 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.362.5.9 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_end
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.362.5.10 `template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.362.5.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 183 of file streambuf.

5.362.5.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 184 of file streambuf.

5.362.5.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

5.362.5.14 `template<typename _CharT, typename _Traits> char_type
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 137 of file fstream.

5.362.5.15 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save
[protected]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

5.362.5.16 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save
[protected]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 139 of file fstream.

5.362.5.17 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::_M_pback_init
[protected]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 140 of file fstream.

5.362.5.18 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::_M_reading
[protected]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

The documentation for this class was generated from the following files:

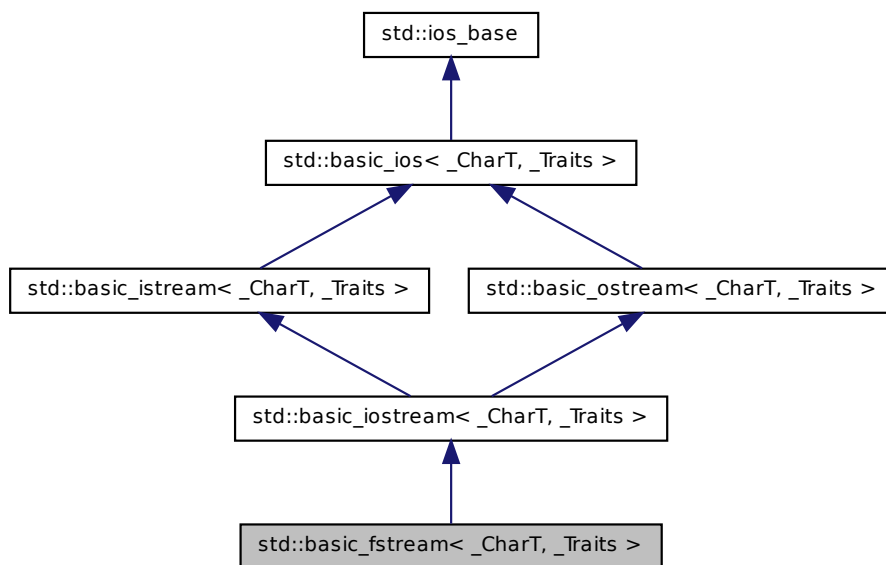
- [fstream](#)
- [fstream.tcc](#)

5.363 `std::basic_fstream< _CharT, _Traits >` Class Template Reference

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_fstream< _CharT, _Traits >`:



Public Types

- typedef [ctype](#)< [_CharT](#) > [__ctype_type](#)
- typedef [ctype](#)< [_CharT](#) > [__ctype_type](#)
- typedef [basic_filebuf](#)< [char_type](#), [traits_type](#) > [__filebuf_type](#)
- typedef [basic_ios](#)< [_CharT](#), [_Traits](#) > [__ios_type](#)
- typedef [basic_ios](#)< [char_type](#), [traits_type](#) > [__ios_type](#)
- typedef [basic_istream](#)< [char_type](#), [traits_type](#) > [__istream_type](#)
- typedef [basic_istream](#)< [_CharT](#), [_Traits](#) > [__istream_type](#)
- typedef [basic_istream](#)< [_CharT](#), [_Traits](#) > [__istream_type](#)
- typedef [num_get](#)< [_CharT](#), [istreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_get_type](#)
- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)
- typedef [basic_ostream](#)< [_CharT](#), [_Traits](#) > [__ostream_type](#)
- typedef [basic_streambuf](#)< [_CharT](#), [_Traits](#) > [__streambuf_type](#)
- typedef [basic_streambuf](#)< [_CharT](#), [_Traits](#) > [__streambuf_type](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_CharT](#) [char_type](#)
- enum [event](#) { [erase_event](#), [imbue_event](#), [copyfmt_event](#) }
- typedef void(* [event_callback](#))(event, [ios_base](#) &, int)
- typedef [_Ios_Fmtflags](#) [fmtflags](#)
- typedef [traits_type::int_type](#) [int_type](#)
- typedef [_Traits::int_type](#) [int_type](#)
- typedef int [io_state](#)
- typedef [_Ios_Iostate](#) [iostate](#)
- typedef [traits_type::off_type](#) [off_type](#)
- typedef [_Traits::off_type](#) [off_type](#)
- typedef int [open_mode](#)
- typedef [_Ios_Openmode](#) [openmode](#)
- typedef [_Traits::pos_type](#) [pos_type](#)
- typedef [traits_type::pos_type](#) [pos_type](#)
- typedef int [seek_dir](#)
- typedef [_Ios_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)

Public Member Functions

- [basic_fstream](#) ()
- [basic_fstream](#) (const char *__s, [ios_base::openmode](#) __mode=[ios_base::in](#)|[ios_base::out](#))
- [basic_fstream](#) (const [std::string](#) &__s, [ios_base::openmode](#) __mode=[ios_base::in](#)|[ios_base::out](#))
- [~basic_fstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- void [close](#) ()
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [__ostream_type](#) & [flush](#) ()
- [streamsize](#) [gcount](#) () const
- template<>
[basic_istream](#)< [wchar_t](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- template<>
[basic_istream](#)< [char](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>
[basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
[basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- template<>
[basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
[basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- bool [is_open](#) ()
- bool [is_open](#) () const

- long & [iword](#) (int __ix)
 - char [narrow](#) (char_type __c, char __dfault) const
 - void [open](#) (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
 - void [open](#) (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
 - [streamsize precision](#) () const
 - [streamsize precision](#) (streamsize __prec)
 - void *& [pword](#) (int __ix)
 - [basic_streambuf](#)<_CharT, _Traits> * [rdbuf](#) ([basic_streambuf](#)<_CharT, _Traits> * __sb)
 - [__filebuf_type](#) * [rdbuf](#) () const
 - [iostate rdstate](#) () const
 - void [register_callback](#) (event_callback __fn, int __index)
 - [__ostream_type](#) & [seekp](#) (pos_type)
 - [__ostream_type](#) & [seekp](#) (off_type, ios_base::seekdir)
 - [fmtflags setf](#) (fmtflags __fmtfl, fmtflags __mask)
 - [fmtflags setf](#) (fmtflags __fmtfl)
 - void [setstate](#) (iostate __state)
 - [pos_type tellp](#) ()
 - [basic_ostream](#)<_CharT, _Traits> * [tie](#) () const
 - [basic_ostream](#)<_CharT, _Traits> * [tie](#) ([basic_ostream](#)<_CharT, _Traits> * __tistr)
 - void [unsetf](#) (fmtflags __mask)
 - char_type [widen](#) (char __c) const
 - [streamsize width](#) (streamsize __wide)
 - [streamsize width](#) () const
-
- [__istream_type](#) & [operator>>](#) (__istream_type &(*__pf)(__istream_type &))
 - [__istream_type](#) & [operator>>](#) (__ios_type &(*__pf)(__ios_type &))
 - [__istream_type](#) & [operator>>](#) (ios_base &(*__pf)(ios_base &))

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`

- [__istream_type](#) & [seekg](#) ([off_type](#), [ios_base::seekdir](#))
- [operator void *](#) () const
- [bool operator!](#) () const
- [__ostream_type](#) & [operator<<](#) ([__ostream_type](#) &(*[__pf](#))([__ostream_type](#) &))
- [__ostream_type](#) & [operator<<](#) ([__ios_type](#) &(*[__pf](#))([__ios_type](#) &))
- [__ostream_type](#) & [operator<<](#) ([ios_base](#) &(*[__pf](#))([ios_base](#) &))

Arithmetic Inserters

All the [operator<<](#) functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type [std::basic_ostream::sentry](#). This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, [ios_base::badbit](#) will be turned on in the stream's error state without causing an [ios_base::failure](#) to be thrown. The original exception will then be rethrown.

- [__ostream_type](#) & [operator<<](#) (long __n)
- [__ostream_type](#) & [operator<<](#) (unsigned long __n)
- [__ostream_type](#) & [operator<<](#) (bool __n)
- [__ostream_type](#) & [operator<<](#) (short __n)
- [__ostream_type](#) & [operator<<](#) (unsigned short __n)
- [__ostream_type](#) & [operator<<](#) (int __n)
- [__ostream_type](#) & [operator<<](#) (unsigned int __n)
- [__ostream_type](#) & [operator<<](#) (long long __n)
- [__ostream_type](#) & [operator<<](#) (unsigned long long __n)
- [__ostream_type](#) & [operator<<](#) (double __f)
- [__ostream_type](#) & [operator<<](#) (float __f)
- [__ostream_type](#) & [operator<<](#) (long double __f)
- [__ostream_type](#) & [operator<<](#) (const void *__p)
- [__ostream_type](#) & [operator<<](#) ([__streambuf_type](#) *__sb)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type [std::basic_ostream::sentry](#). This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, [ios_base::badbit](#) will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosate eofbit`
- static const `iosate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const [locale](#) &__loc)
- void **_M_call_callbacks** ([event](#) __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
 [__istream_type](#) & **_M_extract** (_ValueT &__v)
- [_Words](#) & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & **_M_insert** (_ValueT __v)
- void **init** ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * **_M_callbacks**
- const [__ctype_type](#) * **_M_ctype**
- [iostate](#) **_M_exception**
- [char_type](#) **_M_fill**
- bool **_M_fill_init**
- [fmtflags](#) **_M_flags**
- [streamsize](#) **_M_gcount**
- [locale](#) **_M_ios_locale**
- [_Words](#) **_M_local_word** [[_S_local_word_size](#)]
- const [__num_get_type](#) * **_M_num_get**
- const [__num_put_type](#) * **_M_num_put**
- [streamsize](#) **_M_precision**
- [basic_streambuf](#)< _CharT, _Traits > * **_M_streambuf**
- [iostate](#) **_M_streambuf_state**
- [basic_ostream](#)< _CharT, _Traits > * **_M_tie**
- [streamsize](#) **_M_width**
- [_Words](#) * **_M_word**
- int **_M_word_size**
- [_Words](#) **_M_word_zero**

Friends

- class **sentry**
- class **sentry**

5.363.1 Detailed Description

template<typename _CharT, typename _Traits> class std::basic_fstream< _CharT, _Traits >

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 756 of file `fstream`.

5.363.2 Member Typedef Documentation

5.363.2.1 **template<typename _CharT, typename _Traits> typedef**
ctype<_CharT> std::basic_istream< _CharT, _Traits
>::__ctype_type [inherited]

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file `istream`.

5.363.2.2 **template<typename _CharT, typename _Traits> typedef**
ctype<_CharT> std::basic_ostream< _CharT, _Traits
>::__ctype_type [inherited]

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file `ostream`.

5.363.2.3 **template<typename _CharT, typename _Traits> typedef**
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_istream< _CharT, _Traits >::__num_get_type
[inherited]

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file `istream`.

5.363.2.4 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits >::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream<_CharT, _Traits >](#), [std::basic_ostream< char, _Traits >](#), and [std::basic_ostream< char >](#).

Definition at line 84 of file `basic_ios.h`.

5.363.2.5 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits >::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits >](#).

Definition at line 70 of file `ostream`.

5.363.2.6 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream<_CharT, _Traits >::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits >](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc >](#).

Definition at line 59 of file `istream`.

5.363.2.7 `template<typename _CharT, typename _Traits > typedef _CharT
std::basic_fstream<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits >](#).

Definition at line 760 of file `fstream`.

5.363.2.8 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`
`[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.363.2.9 `typedef _Ios_Fmtflags std::ios_base::fmtflags` `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.363.2.10 `template<typename _CharT, typename _Traits> typedef
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istreamstream< _CharT, _Traits, _Alloc >`.

Definition at line 60 of file `istream`.

5.363.2.11 `template<typename _CharT , typename _Traits > typedef
traits_type::int_type std::basic_fstream< _CharT, _Traits
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_iostream< _CharT, _Traits >`.

Definition at line 762 of file `fstream`.

5.363.2.12 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`

- eofbit
- failbit
- goodbit

Definition at line 338 of file `ios_base.h`.

5.363.2.13 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 62 of file `istream`.

5.363.2.14 `template<typename _CharT , typename _Traits > typedef
traits_type::off_type std::basic_fstream< _CharT, _Traits
>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 764 of file `fstream`.

5.363.2.15 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary

- in
- out
- trunc

Definition at line 369 of file ios_base.h.

5.363.2.16 `template<typename _CharT , typename _Traits > typedef
traits_type::pos_type std::basic_fstream< _CharT, _Traits
>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 763 of file fstream.

5.363.2.17 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 61 of file istream.

5.363.2.18 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

Definition at line 401 of file ios_base.h.

5.363.2.19 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_fstream<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 761 of file `fstream`.

5.363.2.20 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream<_CharT, _Traits>::traits_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 63 of file `istream`.

5.363.3 Member Enumeration Documentation

5.363.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

5.363.4 Constructor & Destructor Documentation

5.363.4.1 `template<typename _CharT, typename _Traits> std::basic_fstream<_CharT, _Traits>::basic_fstream () [inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 783 of file `fstream`.

```
5.363.4.2  template<typename _CharT , typename _Traits >
             std::basic_fstream< _CharT, _Traits >::basic_fstream ( const char *
             __s, ios_base::openmode __mode = ios_base::in | ios_base::out )
             [inline, explicit]
```

Create an input/output file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 796 of file `fstream`.

```
5.363.4.3  template<typename _CharT , typename _Traits >
             std::basic_fstream< _CharT, _Traits >::basic_fstream
             ( const std::string & __s, ios_base::openmode __mode =
             ios_base::in | ios_base::out ) [inline, explicit]
```

Create an input/output file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

Definition at line 811 of file `fstream`.

```
5.363.4.4  template<typename _CharT , typename _Traits >
             std::basic_fstream< _CharT, _Traits >::~~basic_fstream ( )
             [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 826 of file `fstream`.

5.363.5 Member Function Documentation

5.363.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

5.363.5.2 `template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::_M_write (const char_type * __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.363.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.363.5.4 `template<typename _CharT , typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file basic_ios.tcc.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.363.5.5 `template<typename _CharT , typename _Traits > void
std::basic_fstream< _CharT, _Traits >::close () [inline]`

Close the file.

Calls [std::basic_filebuf::close\(\)](#). If that function fails, failbit is set in the stream's error state.

Definition at line 906 of file fstream.

5.363.5.6 `template<typename _CharT , typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.363.5.7 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.363.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions() const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.363.5.9 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions (iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.363.5.10 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail () const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

5.363.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.363.5.12 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill () const [inline,
inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.363.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.363.5.14 fmtflags std::ios_base::flags () const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.363.5.15 template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & std::basic_ostream< _CharT, _Traits>::flush () [inherited]

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

5.363.5.16 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::gcount () const
[inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.363.5.17 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.18 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

`*this`

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.19 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get (char_type * __s, streamsize __n, char_type __delim)`
[inherited]

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.363.5.20 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

5.363.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.363.5.22 `template<typename _CharT, typename _Traits> __istream_type&
 std::basic_istream< _CharT, _Traits >::get (__streambuf_type &
 __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file `istream`.

5.363.5.23 `template<typename _CharT, typename _Traits> __istream_type&
 std::basic_istream< _CharT, _Traits >::getline (char_type * __s,
 streamsize __n) [inline, inherited]`

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

5.363.5.24 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n, char_type __delim)
[inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.363.5.25 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.363.5.26 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::good() const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.363.5.27 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore
(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted

- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.28 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::ignore (streamsize __n) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.363.5.29 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.363.5.30 `template<typename _CharT, typename _Traits> locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.363.5.31 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.363.5.32 `template<typename _CharT, typename _Traits> bool
std::basic_fstream< _CharT, _Traits >::is_open () [inline]`

Wrapper to test for an open file.

Returns

`rdbuf() -> is_open()`

Definition at line 845 of file fstream.

5.363.5.33 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.363.5.34 `template<typename _CharT, typename _Traits> char
std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.363.5.35 `template<typename _CharT, typename _Traits> void
std::basic_fstream<_CharT, _Traits>::open (const char * __s,
ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 866 of file `fstream`.

5.363.5.36 `template<typename _CharT, typename _Traits> void
std::basic_fstream<_CharT, _Traits>::open (const std::string &
__s, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 887 of file `fstream`.

5.363.5.37 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.363.5.38 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.363.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file `ostream`.

5.363.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 180 of file ostream.

```
5.363.5.41  template<typename _CharT, typename _Traits> __ostream_type&
             std::basic_ostream< _CharT, _Traits >::operator<< ( double __f
             ) [inline, inherited]
```

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 209 of file ostream.

```
5.363.5.42  template<typename _CharT , typename _Traits > basic_ostream<
             _CharT, _Traits > & std::basic_ostream< _CharT, _Traits
             >::operator<< ( short __n ) [inherited]
```

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

5.363 std::basic_fstream<_CharT, _Traits> Class Template Reference 1573

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.363.5.43 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 108 of file ostream.

5.363.5.44 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file ostream.

5.363.5.45 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 127 of file ostream.

5.363.5.46 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.363.5.47 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.363.5.48 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.363.5.49 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.363.5.50 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.363.5.51 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.363.5.52 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.363.5.53 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.363.5.54 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.363.5.55 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.363.5.56 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

5.363.5.57 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.363.5.58 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file istream.

5.363.5.59 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

5.363.5.60 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file istream.

5.363.5.61 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 203 of file istream.

5.363.5.62 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned int
& __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 181 of file istream.

5.363.5.63 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 207 of file istream.

5.363.5.64 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file istream.

5.363.5.65 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

5.363.5.66 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (short & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.363.5.67 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.363.5.68 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__istream_type &(*)(__istream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 120 of file istream.

5.363.5.69 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.363.5.70 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.363.5.71 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.363.5.72 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (void *& __p) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

5.363.5.73 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek (void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.363.5.74 streamsize std::ios_base::precision () const [inline, inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

5.363.5.75 streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.363.5.76 template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (char_type __c) [inherited]

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.363.5.77 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

`c` The character to push back into the input stream.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

5.363.5.78 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.363.5.79 `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References `std::basic_ios< _CharT, _Traits >::clear()`.

5.363.5.80 `template<typename _CharT, typename _Traits > __filebuf_type* std::basic_fstream< _CharT, _Traits >::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current [basic_filebuf](#) buffer.

This hides both signatures of [std::basic_ios::rdbuf\(\)](#).

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 837 of file `fstream`.

5.363.5.81 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.363.5.82 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rddbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.363.5.83 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rddbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.363.5.84 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.363.5.85 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

- off* A file offset object.
- dir* The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.86 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]`

Changing the current read position.

Parameters

- pos* A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.87 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.88 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.
dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.363.5.89 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.363.5.90 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios_base::adjustfield](#).

Definition at line 594 of file ios_base.h.

5.363.5.91 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic_ios.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.363.5.92 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.363.5.93 `static bool std::ios_base::sync_with_stdio (bool __sync = true)`
[static, inherited]

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.363.5.94 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (void)` **[inherited]**

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf() -> pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rddbuf()`.

5.363.5.95 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>::pos_type std::basic_ostream< _CharT, _Traits>::tellp()` **[inherited]**

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf() -> pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::rddbuf()`.

5.363.5.96 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>* std::basic_ios< _CharT, _Traits>::tie()` **const [inline, inherited]**

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.363.5.97 `template<typename _CharT, typename _Traits>
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

5.363.5.98 `template<typename _CharT, typename _Traits> basic_istream<
 _CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ungetc (void) [inherited]`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

5.363.5.99 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

5.363.5.100 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()) .widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

5.363.5.101 `streamsize std::ios_base::width (streamsize __wide) [inline,
inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file ios_base.h.

5.363.5.102 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.363.5.103 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed `char` and unsigned `char`.

5.363.5.104 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.363.6 Member Data Documentation

5.363.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.363.6.2 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.363.6.3 `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.363.6.4 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.363.6.5 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.363.6.6 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.363.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.363.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.363.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.363.6.10 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.363.6.11 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.363.6.12 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.363.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.363.6.14 **const iostate std::ios_base::failbit** [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.363.6.15 **const fmtflags std::ios_base::fixed** [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.363.6.16 **const fmtflags std::ios_base::floatfield** [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.363.6.17 **const iostate std::ios_base::goodbit** [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.363.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.363.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.363.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios_base.h.

5.363.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.363.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.363.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.363.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.363.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.363.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.363.6.27 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios_base.h.

5.363.6.28 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios_base.h.

5.363.6.29 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 308 of file ios_base.h.

5.363.6.30 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 389 of file ios_base.h.

5.363.6.31 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file ios_base.h.

5.363.6.32 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following file:

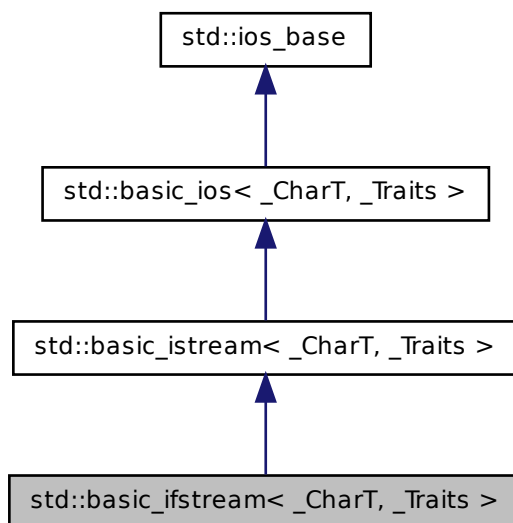
- [fstream](#)

5.364 std::basic_ifstream< _CharT, _Traits > Class Template Reference

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ifstream< _CharT, _Traits >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `traits_type::off_type off_type`

- typedef int **open_mode**
- typedef `_Ios_Openmode` **openmode**
- typedef `traits_type::pos_type` **pos_type**
- typedef int **seek_dir**
- typedef `_Ios_Seekdir` **seekdir**
- typedef `std::streamoff` **streamoff**
- typedef `std::streampos` **streampos**
- typedef `_Traits` **traits_type**
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> __-`
`num_put_type`

Public Member Functions

- `basic_ifstream()`
- `basic_ifstream(const char * __s, ios_base::openmode __mode=ios_base::in)`
- `basic_ifstream(const std::string & __s, ios_base::openmode __mode=ios_base::in)`
- `~basic_ifstream()`
- `const locale & _M_getloc() const`
- `void _M_setstate(iostate __state)`
- `bool bad() const`
- `void clear(iostate __state=goodbit)`
- `void close()`
- `basic_ios & copyfmt(const basic_ios & __rhs)`
- `bool eof() const`
- `iostate exceptions() const`
- `void exceptions(iostate __except)`
- `bool fail() const`
- `char_type fill() const`
- `char_type fill(char_type __ch)`
- `fmtflags flags() const`
- `fmtflags flags(fmtflags __fmtfl)`
- `streamsize gcount() const`
- `template<>`
`basic_istream<char> & getline(char_type * __s, streamsize __n, char_type`
`__delim)`
- `template<>`
`basic_istream<wchar_t> & getline(char_type * __s, streamsize __n, char_type`
`__delim)`
- `locale getloc() const`
- `bool good() const`

- `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
 - `template<>`
`basic_istream< char > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
 - `locale imbue (const locale &__loc)`
 - `bool is_open ()`
 - `bool is_open () const`
 - `long & iword (int __ix)`
 - `char narrow (char_type __c, char __dfault) const`
 - `void open (const char *__s, ios_base::openmode __mode=ios_base::in)`
 - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in)`
 - `streamsize precision () const`
 - `streamsize precision (streamsize __prec)`
 - `void *& pword (int __ix)`
 - `__filebuf_type * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iosate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `void setstate (iosate __state)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width () const`
 - `streamsize width (streamsize __wide)`
-
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
 - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
 - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several

effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, [`ios_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`__istream_type & operator>> \(bool &__n\)`](#)
- [`__istream_type & operator>> \(short &__n\)`](#)
- [`__istream_type & operator>> \(unsigned short &__n\)`](#)
- [`__istream_type & operator>> \(int &__n\)`](#)
- [`__istream_type & operator>> \(unsigned int &__n\)`](#)
- [`__istream_type & operator>> \(long &__n\)`](#)
- [`__istream_type & operator>> \(unsigned long &__n\)`](#)
- [`__istream_type & operator>> \(long long &__n\)`](#)
- [`__istream_type & operator>> \(unsigned long long &__n\)`](#)
- [`__istream_type & operator>> \(float &__f\)`](#)
- [`__istream_type & operator>> \(double &__f\)`](#)
- [`__istream_type & operator>> \(long double &__f\)`](#)
- [`__istream_type & operator>> \(void *&__p\)`](#)
- [`__istream_type & operator>> \(__streambuf_type *__sb\)`](#)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type [`std::basic_istream::sentry`](#) with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by [`gcount\(\)`](#).

If an exception is thrown during extraction, [`ios_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`int_type get \(\)`](#)
- [`__istream_type & get \(char_type &__c\)`](#)
- [`__istream_type & get \(char_type *__s, streamsize __n, char_type __delim\)`](#)
- [`__istream_type & get \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & get \(__streambuf_type &__sb, char_type __delim\)`](#)
- [`__istream_type & get \(__streambuf_type &__sb\)`](#)
- [`__istream_type & getline \(char_type *__s, streamsize __n, char_type __delim\)`](#)
- [`__istream_type & getline \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & ignore \(\)`](#)
- [`__istream_type & ignore \(streamsize __n\)`](#)

- [__istream_type](#) & ignore (streamsize __n, int_type __delim)
- [int_type](#) peek ()
- [__istream_type](#) & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- [__istream_type](#) & putback (char_type __c)
- [__istream_type](#) & unget ()
- int sync ()
- [pos_type](#) tellg ()
- [__istream_type](#) & seekg (pos_type)
- [__istream_type](#) & seekg (off_type, ios_base::seekdir)
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iostate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out

- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- template<typename _ValueT >
[__istream_type](#) & [_M_extract](#) (_ValueT &__v)
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * [_M_callbacks](#)
- const [__ctype_type](#) * [_M_ctype](#)
- [iostate](#) [_M_exception](#)
- [char_type](#) [_M_fill](#)
- bool [_M_fill_init](#)
- [fmtflags](#) [_M_flags](#)
- [streamsize](#) [_M_gcount](#)
- [locale](#) [_M_ios_locale](#)
- [_Words](#) [_M_local_word](#) [[_S_local_word_size](#)]
- const [__num_get_type](#) * [_M_num_get](#)
- const [__num_put_type](#) * [_M_num_put](#)
- [streamsize](#) [_M_precision](#)
- [basic_streambuf](#)< _CharT, _Traits > * [_M_streambuf](#)
- [iostate](#) [_M_streambuf_state](#)

- [basic_ostream](#)< _CharT, _Traits > * _M_tie
- [streamsize](#) _M_width
- _Words * _M_word
- int _M_word_size
- _Words _M_word_zero

Friends

- class `sentry`

5.364.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ifstream< _CharT, _Traits >`

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 415 of file `fstream`.

5.364.2 Member Typedef Documentation

5.364.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_istream< _CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file `istream`.

5.364.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_istream< _CharT, _Traits >::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file `istream`.

5.364.2.3 `template<typename _CharT, typename _Traits> typedef
 num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
 > std::basic_ios<_CharT, _Traits>::__num_put_type
 [inherited]`

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

5.364.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
 std::basic_ifstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits>`.

Definition at line 419 of file `fstream`.

5.364.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
 [inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.364.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file ios_base.h.

5.364.2.7 `template<typename _CharT , typename _Traits > typedef
traits_type::int_type std::basic_ifstream< _CharT, _Traits
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 421 of file fstream.

5.364.2.8 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.364.2.9 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_ifstream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits>`.

Definition at line 423 of file `fstream`.

5.364.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

5.364.2.11 `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_ifstream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 422 of file `fstream`.

5.364.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

5.364.2.13 `template<typename _CharT , typename _Traits > typedef _Traits std::basic_ifstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 420 of file `fstream`.

5.364.3 Member Enumeration Documentation

5.364.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

5.364.4 Constructor & Destructor Documentation

5.364.4.1 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream ()
[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 441 of file `fstream`.

5.364.4.2 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (const char
* __s, ios_base::openmode __mode = ios_base::in) [inline,
explicit]`

Create an input file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::in](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 455 of file `fstream`.

5.364.4.3 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (const
std::string & __s, ios_base::openmode __mode = ios_base::in)
[inline, explicit]`

Create an input file stream.

Parameters

s `std::string` specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::in](#) is automatically included in *mode*.

Definition at line 471 of file `fstream`.

5.364.4.4 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream ()
[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 486 of file fstream.

5.364.5 Member Function Documentation

5.364.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.364.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.364.5.3 `template<typename _CharT , typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.364.5.4 `template<typename _CharT , typename _Traits > void
std::basic_ifstream< _CharT, _Traits >::close () [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 564 of file `fstream`.

5.364.5.5 `template<typename _CharT , typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.364.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.364.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions() const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.364.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions(iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.364.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail() const` [`inline`, `inherited`]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.364.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill(char_type __ch)` [`inline`, `inherited`]

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.364.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill () const [inline,
inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.364.5.12 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

5.364.5.13 fmtflags std::ios_base::flags () const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.364.5.14 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline, inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.364.5.15 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.364.5.16 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.364.5.17 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n, char_type __delim) [inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored

- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.364.5.18 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

`*this`

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

5.364.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file istream.

5.364.5.20 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

5.364.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file istream.

Referenced by `std::basic_istream<char>::getline()`.

5.364.5.22 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored

3. $n-1$ characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sputc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.364.5.23 locale `std::ios_base::getloc () const` [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.364.5.24 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good () const` [inline, inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file basic_ios.h.

5.364.5.25 `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::ignore(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_ifstream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.364.5.26 `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::ignore(streamsize __n) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.364.5.27 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.364.5.28 `template<typename _CharT, typename _Traits > locale std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

5.364 std::basic_ifstream< _CharT, _Traits > Class Template Reference 1631

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References [std::ios_base::getloc\(\)](#), and [std::basic_ios< _CharT, _Traits >::rdbuf\(\)](#).

Referenced by [std::operator<<\(\)](#).

5.364.5.29 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References [std::ios_base::badbit](#), and [std::ios_base::goodbit](#).

5.364.5.30 `template<typename _CharT , typename _Traits > bool
std::basic_ifstream< _CharT, _Traits >::is_open () [inline]`

Wrapper to test for an open file.

Returns

`rdbuf() -> is_open()`

Definition at line 505 of file fstream.

5.364.5.31 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.364.5.32 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()) .narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.364.5.33 `template<typename _CharT , typename _Traits > void
std::basic_ifstream<_CharT, _Traits>::open (const std::string &
__s, ios_base::openmode __mode = ios_base::in) [inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 546 of file `fstream`.

5.364.5.34 `template<typename _CharT, typename _Traits> void
std::basic_ifstream< _CharT, _Traits >::open (const char * __s,
ios_base::openmode __mode = ios_base::in) [inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 526 of file `fstream`.

5.364.5.35 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits >::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.364.5.36 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.364.5.37 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

5.364.5.38 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.364.5.39 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.364.5.40 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

```
5.364.5.41  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_ifstream<_CharT, _Traits>::operator>> ( float & __f
             ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

```
5.364.5.42  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_ifstream<_CharT, _Traits>::operator>> ( double &
             __f ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

```
5.364.5.43  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_ifstream<_CharT, _Traits>::operator>> ( bool & __n
             ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 167 of file istream.

```
5.364.5.44  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_istream< _CharT, _Traits >::operator>> ( long double
             & __f ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 211 of file istream.

```
5.364.5.45  template<typename _CharT , typename _Traits > basic_istream<
             _CharT, _Traits > & std::basic_istream< _CharT, _Traits
             >::operator>> ( short & __n ) [inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

5.364.5.46 `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream< _CharT, _Traits >::operator>> (void *& __p) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file istream.

5.364.5.47 `template<typename _CharT , typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.364.5.48 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.364.5.49 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

5.364.5.50 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.364.5.51 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.364.5.52 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

```
5.364.5.53  template<typename _CharT, typename _Traits> __istream_type&
              std::basic_istream< _CharT, _Traits >::operator>> ( long & __n
              ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file `istream`.

```
5.364.5.54  template<typename _CharT , typename _Traits > basic_istream<
              _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
              >::peek ( void ) [inherited]
```

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

```
5.364.5.55  streamsize std::ios_base::precision ( streamsize __prec )
              [inline, inherited]
```

Changing flags.

5.364 std::basic_ifstream<_CharT, _Traits> Class Template Reference 1641

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.364.5.56 streamsize std::ios_base::precision () const [inline, inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), and std::operator<<().

5.364.5.57 template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inherited]

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.364.5.58 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.364.5.59 `template<typename _CharT, typename _Traits > __filebuf_type* std::basic_ifstream< _CharT, _Traits >::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 497 of file `fstream`.

5.364.5.60 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.364.5.61 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate () const [inline, inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.364.5.62 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.364.5.63 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream<_CharT, _Traits>::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets `eofbit` and extracts no characters
- if *A* == 0, extracts no characters

- if $A > 0$, extracts $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_ifstream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.364.5.64 void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn The function to add.

__index The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.364.5.65 template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.364.5.66 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff (off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.364.5.67 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by std::boolalpha(), std::dec(), std::fixed(), std::hex(), std::internal(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

5.364.5.68 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios_base::adjustfield](#).

Definition at line 594 of file ios_base.h.

5.364.5.69 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate __state) [inline, inherited]

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(),

`std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.364.5.70 `template<typename _CharT, typename _Traits> int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.364.5.71 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.364.5.72 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg(void) [inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rddbuf()`.

5.364.5.73 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.364.5.74 `template<typename _CharT, typename _Traits>
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits
 >::tie (basic_ostream<_CharT, _Traits> * __tistr) [inline,
 inherited]`

Ties this stream to an output stream.

Parameters

tistr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

5.364.5.75 `template<typename _CharT, typename _Traits> basic_istream<
 _CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
 (void) [inherited]`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

5.364.5.76 `void std::ios_base::unsetf (fmtflags __mask) [inline,
 inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

5.364.5.77 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

5.364.5.78 `streamsize std::ios_base::width () const [inline,
inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), std::num_put<_CharT, _OutIter>::do_put(), and std::operator>>().

5.364.5.79 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file `ios_base.h`.

5.364.5.80 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.364.6 Member Data Documentation

5.364.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.364.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.364.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.364.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.364.6.5 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.364.6.6 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.364.6.7 `const seekdir std::ios_base::beg` **[static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

5.364.6.8 `const openmode std::ios_base::binary` **[static, inherited]**

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.364.6.9 `const fmtflags std::ios_base::boolalpha` **[static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.364.6.10 `const seekdir std::ios_base::cur` **[static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.364.6.11 `const fmtflags std::ios_base::dec` **[static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

5.364.6.12 `const seekdir std::ios_base::end` **[static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.364.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.364.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.364.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.364.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.364.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.364.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.364.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.364.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios_base.h.

5.364.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.364.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.364.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.364.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.364.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.364.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.364.6.27 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios_base.h.

5.364.6.28 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios_base.h.

5.364.6.29 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 308 of file ios_base.h.

5.364.6.30 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 389 of file ios_base.h.

5.364.6.31 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file ios_base.h.

5.364.6.32 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following file:

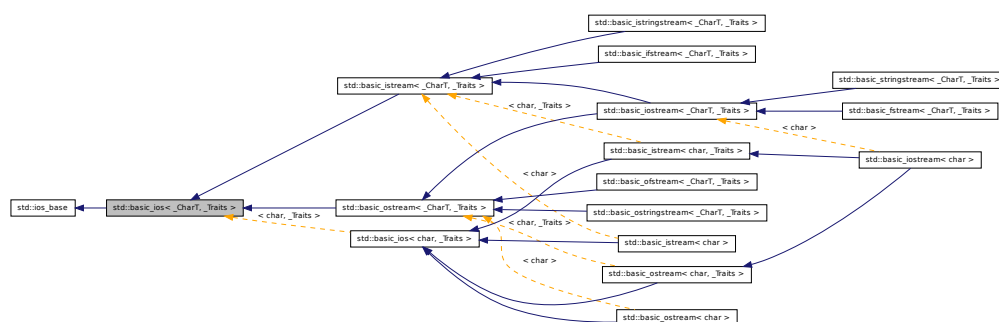
- [fstream](#)

5.365 std::basic_ios< _CharT, _Traits > Class Template Reference

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Inheritance diagram for `std::basic_ios< _CharT, _Traits >`:



Public Types

- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `int` `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `int` `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_CharT` `char_type`
- typedef `_Traits::int_type` `int_type`
- typedef `_Traits::pos_type` `pos_type`
- typedef `_Traits::off_type` `off_type`
- typedef `_Traits` `traits_type`
- typedef `ctype< _CharT >` `__ctype_type`

- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)
- typedef [num_get](#)< [_CharT](#), [istreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_get_type](#)

Public Member Functions

- [basic_ios](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *[__sb](#))
- virtual [~basic_ios](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) [__state](#))
- bool [bad](#) () const
- void [clear](#) ([iostate](#) [__state](#)=goodbit)
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &[__rhs](#))
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) [__except](#))
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) [__ch](#))
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) [__fmtfl](#))
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &[__loc](#))
- long & [iword](#) (int [__ix](#))
- [char](#) [narrow](#) ([char_type](#) [__c](#), [char](#) [__dfault](#)) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) [__prec](#))
- void *& [pword](#) (int [__ix](#))
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *[__sb](#))
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register_callback](#) ([event_callback](#) [__fn](#), int [__index](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) [__fmtfl](#), [fmtflags](#) [__mask](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) [__fmtfl](#))
- void [setstate](#) ([iostate](#) [__state](#))
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) ([basic_ostream](#)< [_CharT](#), [_Traits](#) > *[__tiestr](#))
- void [unsetf](#) ([fmtflags](#) [__mask](#))
- [char_type](#) [widen](#) ([char](#) [__c](#)) const

- [streamsize width](#) () const
- [streamsize width](#) (streamsize __wide)
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags adjustfield](#)
- static const [openmode app](#)
- static const [openmode ate](#)
- static const [iostate badbit](#)
- static const [fmtflags basefield](#)
- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- `basic_ios` ()
- void `_M_cache_locale` (const `locale` &__loc)
- void `_M_call_callbacks` (`event` __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int __index, bool __iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > *__sb)

Protected Attributes

- `_Callback_list` * `_M_callbacks`
- const `__ctype_type` * `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` * `_M_num_get`
- const `__num_put_type` * `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > * `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > * `_M_tie`
- `streamsize` `_M_width`
- `_Words` * `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

5.365.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ios< _CharT,
_Traits >
```

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 62 of file `basic_ios.h`.

5.365.2 Member Typedef Documentation

5.365.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, `std::basic_ostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 82 of file `basic_ios.h`.

5.365.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>
std::basic_ios<_CharT, _Traits>::__num_get_type`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, and `std::basic_istream<char>`.

Definition at line 86 of file `basic_ios.h`.

5.365.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>
std::basic_ios<_CharT, _Traits>::__num_put_type`

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

5.365.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_ios<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istream< char, _Traits >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 71 of file `basic_ios.h`.

5.365.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` **[inherited]**

The type of an event callback function.

Parameters

- event* One of the members of the event enum.
- ios_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.365.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`

- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.365.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 72 of file `basic_ios.h`.

5.365.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 338 of file ios_base.h.

5.365.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 74 of file basic_ios.h.

5.365.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 369 of file ios_base.h.

5.365.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 73 of file `basic_ios.h`.

5.365.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

5.365.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 75 of file basic_ios.h.

5.365.3 Member Enumeration Documentation

5.365.3.1 enum std::ios_base::event [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 427 of file ios_base.h.

5.365.4 Constructor & Destructor Documentation

5.365.4.1 template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios(basic_streambuf<_CharT, _Traits> * __sb) [inline, explicit]

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 260 of file basic_ios.h.

5.365.4.2 template<typename _CharT, typename _Traits> virtual std::basic_ios<_CharT, _Traits>::~~basic_ios() [inline, virtual]

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by [rdbuf\(\)](#).

Definition at line 272 of file basic_ios.h.

5.365.4.3 template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios() [inline, protected]

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 450 of file basic_ios.h.

5.365.5 Member Function Documentation

5.365.5.1 `const locale& std::ios_base::_M_getloc () const` [`inline`, `inherited`]

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.365.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const` [`inline`]

Fast error checking.

Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.365.5.3 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.365.5.4 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs)`

Copies fields of `__rhs` into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.365.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.365.5.6 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions() const [inline]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.365.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions (iostate __except) [inline]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.365.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail () const [inline]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.365.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill () const [inline]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.365.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.365.5.11 fmtflags std::ios_base::flags () const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.365.5.12 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline, inherited]

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.365.5.13 locale std::ios_base::getloc () const [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If imbue(loc) has previously been called, then this function returns loc. Otherwise, it returns a copy of std::locale(), the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Traits >::imbue(), std::operator>>(), and std::ws().

5.365.5.14 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.365.5.15 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.365.5.16 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.365.5.17 long& std::ios_base::iword (int __ix) [inline, inherited]

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios_base.h.

**5.365.5.18 template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline]**

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of *char_type* to a character of *char*, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file basic_ios.h.

5.365.5.19 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * () const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.365.5.20 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! () const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.365.5.21 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.365.5.22 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file ios_base.h.

5.365.5.23 void*& std::ios_base::pword (int __ix) [inline, inherited]

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.365.5.24 template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb)

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.365.5.25 `template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,
_Traits>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), [std::basic_ofstream<_CharT, _Traits>](#), [std::basic_fstream<_CharT, _Traits>](#), [std::basic_istream<_CharT, _Traits, _Alloc>](#), [std::basic_ostringstream<_CharT, _Traits, _Alloc>](#), and [std::basic_stringstream<_CharT, _Traits, _Alloc>](#).

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

5.365.5.26 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate () const [inline]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.365.5.27 void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn The function to add.

__index The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.365.5.28 fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]

Setting new format flags.

Parameters

__fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by std::boolalpha(), std::dec(), std::fixed(), std::hex(), std::internal(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

5.365.5.29 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]

Setting new format flags.

Parameters

__fmtfl Additional flags to set.

__mask The flags mask for *__fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.365.5.30 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.365.5.31 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.365.5.32 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file basic_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.365.5.33 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file basic_ios.h.

5.365.5.34 `void std::ios_base::unsetf(fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

5.365.5.35 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::widen (char __c) const
[inline]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

5.365.5.36 `streamsize std::ios_base::width () const [inline,
inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), std::num_put<_CharT, _OutIter>::do_put(), and std::operator>>().

5.365.5.37 streamsize std::ios_base::width (streamsize __wide) [inline, inherited]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file ios_base.h.

5.365.5.38 static int std::ios_base::xalloc () throw () [static, inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.365.6 Member Data Documentation**5.365.6.1 const fmtflags std::ios_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file ios_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.365.6.2 const openmode std::ios_base::app [static, inherited]

Seek to end before each write.

Definition at line 372 of file ios_base.h.

5.365.6.3 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.365.6.4 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.365.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.365.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.365.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.365.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.365.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.365.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.365.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.365.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.365.6.13 **const iostate std::ios_base::failbit** [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.365.6.14 **const fmtflags std::ios_base::fixed** [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.365.6.15 **const fmtflags std::ios_base::floatfield** [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.365.6.16 **const iostate std::ios_base::goodbit** [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.365.6.17 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.365.6.18 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.365.6.19 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios_base.h.

5.365.6.20 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.365.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.365.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.365.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.365.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.365.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.365.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios_base.h.

5.365.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

5.366 `std::basic_iostream< _CharT, _Traits >` Class Template Reference 1689

Definition at line 305 of file `ios_base.h`.

5.365.6.28 `const fmtflags std::ios_base::skipws` [static, inherited]

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.365.6.29 `const openmode std::ios_base::trunc` [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.365.6.30 `const fmtflags std::ios_base::unitbuf` [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.365.6.31 `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following files:

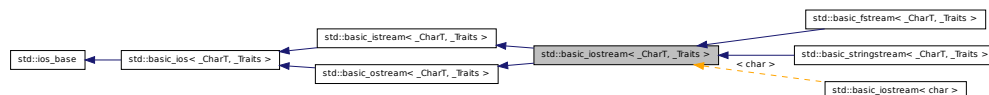
- [basic_ios.h](#)
- [basic_ios.tcc](#)

5.366 `std::basic_iostream< _CharT, _Traits >` Class Template Reference

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Inheritance diagram for `std::basic_istream<_CharT, _Traits>`:



Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef `_Traits::int_type` `int_type`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef `_Traits::off_type` `off_type`
- typedef `int` `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef `_Traits::pos_type` `pos_type`
- typedef `int` `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`

- typedef [std::streampos](#) **streampos**
- typedef [_Traits traits_type](#)
- typedef [_Traits traits_type](#)
- typedef [num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __-](#)
[num_put_type](#)

Public Member Functions

- [basic_iostream](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *__sb)
- virtual [~basic_iostream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type fill](#) () const
- [char_type fill](#) ([char_type](#) __ch)
- [fmtflags flags](#) ([fmtflags](#) __fmtfl)
- [fmtflags flags](#) () const
- [__ostream_type](#) & [flush](#) ()
- [streamsize gcount](#) () const
- template<>
[basic_istream](#)< [char](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#)
__delim)
- template<>
[basic_istream](#)< [wchar_t](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#)
__delim)
- [locale getloc](#) () const
- bool [good](#) () const
- template<>
[basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
[basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- template<>
[basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
[basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- [locale imbue](#) (const [locale](#) &__loc)

- `long & iword (int __ix)`
 - `char narrow (char_type __c, char __dfault) const`
 - `streamsize precision () const`
 - `streamsize precision (streamsize __prec)`
 - `void *& pword (int __ix)`
 - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iostate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `__ostream_type & seekp (pos_type)`
 - `__ostream_type & seekp (off_type, ios_base::seekdir)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `void setstate (iostate __state)`
 - `pos_type tellp ()`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width (streamsize __wide)`
 - `streamsize width () const`
-
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
 - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
 - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`

- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`

- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void **_M_cache_locale** (const [locale](#) &__loc)
- void **_M_call_callbacks** ([event](#) __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
 [__istream_type](#) & **_M_extract** (_ValueT &__v)
- [_Words](#) & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & **_M_insert** (_ValueT __v)
- void **init** ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * **_M_callbacks**
- const [__ctype_type](#) * **_M_ctype**
- [iostate](#) **_M_exception**
- [char_type](#) **_M_fill**
- bool **_M_fill_init**
- [fmtflags](#) **_M_flags**
- [streamsize](#) **_M_gcount**
- [locale](#) **_M_ios_locale**
- [_Words](#) **_M_local_word** [[_S_local_word_size](#)]
- const [__num_get_type](#) * **_M_num_get**
- const [__num_put_type](#) * **_M_num_put**
- [streamsize](#) **_M_precision**
- [basic_streambuf](#)< _CharT, _Traits > * **_M_streambuf**
- [iostate](#) **_M_streambuf_state**
- [basic_ostream](#)< _CharT, _Traits > * **_M_tie**
- [streamsize](#) **_M_width**
- [_Words](#) * **_M_word**
- int **_M_word_size**
- [_Words](#) **_M_word_zero**

Friends

- class **sentry**
- class **sentry**

5.366.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _-
CharT, _Traits >
```

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 769 of file istream.

5.366.2 Member Typedef Documentation

5.366.2.1 `template<typename _CharT, typename _Traits> typedef`
`ctype<_CharT> std::basic_istream< _CharT, _Traits`
`>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file istream.

5.366.2.2 `template<typename _CharT, typename _Traits> typedef`
`ctype<_CharT> std::basic_ostream< _CharT, _Traits`
`>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file ostream.

5.366.2.3 `template<typename _CharT, typename _Traits> typedef`
`num_get<_CharT, istreambuf_iterator<_CharT, _Traits>`
`> std::basic_istream< _CharT, _Traits >::__num_get_type`
`[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file istream.

5.366.2.4 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file ostream.

5.366.2.5 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream<_CharT, _Traits>](#), [std::basic_ostream<char, _Traits>](#), and [std::basic_ostream<char>](#).

Definition at line 84 of file basic_ios.h.

5.366.2.6 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 59 of file istream.

5.366.2.7 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_iostream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Reimplemented in [std::basic_fstream<_CharT, _Traits>](#), and [std::basic_stringstream<_CharT, _Traits, _Alloc>](#).

Definition at line 777 of file `istream`.

5.366.2.8 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` [`inherited`]

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.366.2.9 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [`inherited`]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`

- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file ios_base.h.

5.366.2.10 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_iostream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 778 of file istream.

5.366.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_istream< _CharT, _Traits >::int_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 60 of file istream.

5.366.2.12 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.366.2.13 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_iostream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 780 of file `istream`.

5.366.2.14 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istream< _CharT, _Traits, _Alloc >`.

Definition at line 62 of file `istream`.

5.366.2.15 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`

- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

5.366.2.16 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_iostream< _CharT, _Traits
>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 779 of file `istream`.

5.366.2.17 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 61 of file `istream`.

5.366.2.18 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`

- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

Definition at line 401 of file ios_base.h.

5.366.2.19 `template<typename _CharT, typename _Traits> typedef
_Traits std::basic_istream< _CharT, _Traits >::traits_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istream< _CharT, _Traits, _Alloc >](#).

Definition at line 63 of file istream.

5.366.2.20 `template<typename _CharT, typename _Traits> typedef _Traits
std::basic_iostream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Reimplemented in [std::basic_fstream< _CharT, _Traits >](#), and [std::basic_stringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 781 of file istream.

5.366.3 Member Enumeration Documentation

5.366.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 427 of file ios_base.h.

5.366.4 Constructor & Destructor Documentation

5.366.4.1 `template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
basic_streambuf< _CharT, _Traits > * __sb) [inline,
explicit]`

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 794 of file istream.

5.366.4.2 `template<typename _CharT, typename _Traits> virtual
std::basic_istream< _CharT, _Traits >::~~basic_istream ()
[inline, virtual]`

Destructor does nothing.

Definition at line 801 of file istream.

5.366.5 Member Function Documentation

5.366.5.1 `const locale& std::ios_base::_M_getloc () const [inline,
inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.366.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream< _CharT, _Traits >::_M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.366.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.366.5.4 `template<typename _CharT , typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.366.5.5 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.366.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.366.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions (iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file basic_ios.h.

5.366.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions () const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.366.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.366.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill(char_type __ch) [inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.366.5.11 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.366.5.12 fmtflags std::ios_base::flags() const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.366.5.13 fmtflags std::ios_base::flags(fmtflags __fmtfl) [inline, inherited]

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.366.5.14 template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush() [inherited]

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.366.5.15 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream<_CharT, _Traits>::gcount () const
[inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file `istream`.

5.366.5.16 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits
>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.17 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n, char_type __delim) [inherited]`

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.366.5.18 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.366.5.20 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

5.366.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

5.366.5.22 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.366.5.23 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

5.366.5.24 locale std::ios_base::getloc () const [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

5.366.5.25 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good () const [inline, inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.366.5.26 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.27 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

5.366 std::basic_istream< _CharT, _Traits > Class Template Reference 1717

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.366.5.28 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (streamsize __n) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.366.5.29 template<typename _CharT, typename _Traits > locale std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls ios_base::imbue(loc), and if a stream buffer is associated with this stream, calls that buffer's pubimbue(loc).

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.366.5.30 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.366.5.31 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.366.5.32 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.366.5.33 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

5.366.5.34 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

5.366.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (bool __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.366.5.36 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.366.5.37 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.366.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

5.366.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

5.366.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

5.366.5.41 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (short __n) [inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.366.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.366.5.43 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.366.5.44 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long long
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.366.5.45 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.366.5.46 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file `ostream`.

5.366.5.47 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file `ostream`.

5.366.5.48 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file `ostream`.

5.366.5.49 `template<typename _CharT, typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.50 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.366.5.51 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.366.5.52 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (void *& __p
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

5.366.5.53 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline,
inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file `istream`.

5.366.5.54 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.366.5.55 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.56 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

5.366.5.57 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file istream.

5.366.5.58 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 185 of file istream.

5.366.5.59 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 167 of file istream.

5.366.5.60 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 189 of file istream.

5.366.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (short & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.62 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.366.5.63 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file istream.

5.366.5.64 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file istream.

5.366.5.65 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (float & __f
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.366.5.66 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits
>::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.67 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (long double
& __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

5.366.5.68 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

5.366 std::basic_istream< _CharT, _Traits > Class Template Reference 1733

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.366.5.69 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits>::int_type std::basic_istream< _CharT, _Traits
>::peek(void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

5.366.5.70 `streamsize std::ios_base::precision(streamsize __prec)
[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file ios_base.h.

5.366.5.71 `streamsize std::ios_base::precision() const [inline,
inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits >::copyfmt(), and std::operator<<().

5.366.5.72 `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::put (char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios<_CharT, _Traits >::rdbuf(), and std::basic_ios<_CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

5.366.5.73 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.366.5.74 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.366.5.75 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::rdbuf () const [inline, inherited]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

5.366.5.76 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.366.5.77 `template<typename _CharT, typename _Traits> istate
std::basic_ios< _CharT, _Traits >::rdstate () const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.366.5.78 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.79 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits>::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

5.366.5.80 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.366.5.81 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.82 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.83 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.84 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

`*this`

5.366 std::basic_iostream< _CharT, _Traits > Class Template Reference 1741

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.85 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.366.5.86 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.366.5.87 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.366.5.88 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

5.366 std::basic_iostream< _CharT, _Traits > Class Template Reference 1743

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.366.5.89 static bool std::ios_base::sync_with_stdio (bool __sync = true) [static, inherited]

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.366.5.90 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tell (void) [inherited]

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

5.366.5.91 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.366.5.92 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits>* __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

5.366.5.93 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.366.5.94 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget
(void) [inherited]`

Unextracting the previous character.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

5.366.5.95 `void std::ios_base::unsetf(fmtflags __mask) [inline,
inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.366.5.96 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()) .widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.366.5.97 `streamsize std::ios_base::width () const [inline,
inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

5.366.5.98 `streamsize std::ios_base::width (streamsize __wide) [inline,
inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.366.5.99 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.366.5.100 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.366.6 Member Data Documentation

5.366.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount` [**protected**, **inherited**]

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.366.6.2 `const fmtflags std::ios_base::adjustfield` [**static**, **inherited**]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.366.6.3 `const openmode std::ios_base::app` [**static**, **inherited**]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.366.6.4 `const openmode std::ios_base::ate` [**static**, **inherited**]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.366.6.5 `const iostate std::ios_base::badbit` [**static**, **inherited**]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.366.6.6 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.366.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.366.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.366.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.366.6.10 `const seekdir std::ios_base::cur` [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.366.6.11 `const fmtflags std::ios_base::dec` [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

5.366.6.12 `const seekdir std::ios_base::end` [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.366.6.13 `const iostate std::ios_base::eofbit` [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

5.366.6.14 `const iostate std::ios_base::failbit` [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.366.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.366.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of set f.

Definition at line 324 of file ios_base.h.

5.366.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.366.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.366.6.19 `const openmode std::ios_base::in` **[static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.366.6.20 `const fmtflags std::ios_base::internal` **[static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

5.366.6.21 `const fmtflags std::ios_base::left` **[static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.366.6.22 `const fmtflags std::ios_base::oct` **[static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.366.6.23 `const openmode std::ios_base::out` **[static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

5.366 std::basic_iostream< _CharT, _Traits > Class Template Reference 1753

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.366.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file `ios_base.h`.

5.366.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

5.366.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file `ios_base.h`.

5.366.6.27 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file `ios_base.h`.

5.366.6.28 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

5.366.6.29 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.366.6.30 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 389 of file ios_base.h.

5.366.6.31 `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 311 of file ios_base.h.

5.366.6.32 `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

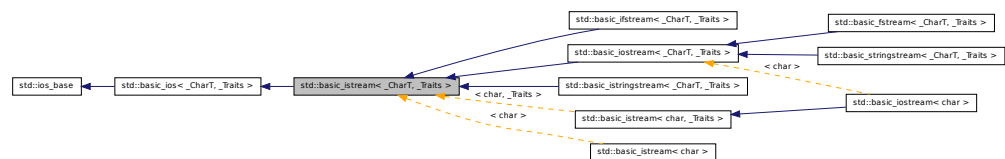
- [istream](#)

5.367 `std::basic_istream< _CharT, _Traits >` Class Template Reference

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual input.

Inheritance diagram for `std::basic_istream< _CharT, _Traits >`:



Classes

- class [sentry](#)

Performs setup work for input streams.

Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(* `event_callback`)(`event`, `ios_base` &, int)
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef int `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef int `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`

- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

Public Member Functions

- `basic_istream` (`__streambuf_type` *__sb)
- virtual `~basic_istream` ()
- const `locale` & `_M_getloc` () const
- void `_M_setstate` (`iostate` __state)
- bool `bad` () const
- void `clear` (`iostate` __state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &__rhs)
- bool `eof` () const
- `iostate` `exceptions` () const

- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [fmtflags](#) [flags](#) () const
- [streamsize](#) [gcount](#) () const
- template<>
 [basic_istream](#)< [char](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- template<>
 [basic_istream](#)< [wchar_t](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>
 [basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
 [basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- template<>
 [basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
 [basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- long & [iword](#) (int __ix)
- [char](#) [narrow](#) ([char_type](#) __c, [char](#) __dfault) const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- [streamsize](#) [precision](#) () const
- void *& [pword](#) (int __ix)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > * __sb)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register_callback](#) ([event_callback](#) __fn, int __index)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
- void [setstate](#) ([iostate](#) __state)
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) ([basic_ostream](#)< [_CharT](#), [_Traits](#) > * __tistr)
- void [unsetf](#) ([fmtflags](#) __mask)
- [char_type](#) [widen](#) ([char](#) __c) const
- [streamsize](#) [width](#) ([streamsize](#) __wide)

- `streamsize width () const`
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [int_type](#) [get](#) ()
- [__istream_type](#) & [get](#) ([char_type](#) &__c)
- [__istream_type](#) & [get](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [__istream_type](#) & [get](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [get](#) ([__streambuf_type](#) &__sb, [char_type](#) __delim)
- [__istream_type](#) & [get](#) ([__streambuf_type](#) &__sb)
- [__istream_type](#) & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [__istream_type](#) & [getline](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [ignore](#) ()
- [__istream_type](#) & [ignore](#) ([streamsize](#) __n)
- [__istream_type](#) & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- [int_type](#) [peek](#) ()
- [__istream_type](#) & [read](#) ([char_type](#) *__s, [streamsize](#) __n)
- [streamsize](#) [readsome](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [putback](#) ([char_type](#) __c)
- [__istream_type](#) & [unget](#) ()
- [int](#) [sync](#) ()
- [pos_type](#) [tellg](#) ()
- [__istream_type](#) & [seekg](#) ([pos_type](#))
- [__istream_type](#) & [seekg](#) ([off_type](#), [ios_base::seekdir](#))
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static [bool](#) [sync_with_stdio](#) ([bool](#) __sync=true)
- static [int](#) [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)

- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- template<typename _ValueT >
[__istream_type](#) & [_M_extract](#) (_ValueT &__v)
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * [_M_callbacks](#)
- const [__ctype_type](#) * [_M_ctype](#)
- [iostate](#) [_M_exception](#)
- [char_type](#) [_M_fill](#)
- bool [_M_fill_init](#)

- [fmtflags](#) [_M_flags](#)
- [streamsize](#) [_M_gcount](#)
- [locale](#) [_M_ios_locale](#)
- [_Words](#) [_M_local_word](#) [[_S_local_word_size](#)]
- [const](#) [__num_get_type](#) * [_M_num_get](#)
- [const](#) [__num_put_type](#) * [_M_num_put](#)
- [streamsize](#) [_M_precision](#)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [_M_streambuf](#)
- [iostate](#) [_M_streambuf_state](#)
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [_M_tie](#)
- [streamsize](#) [_M_width](#)
- [_Words](#) * [_M_word](#)
- [int](#) [_M_word_size](#)
- [_Words](#) [_M_word_zero](#)

Friends

- [class](#) [sentry](#)

5.367.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_istream< _CharT, _Traits >`

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual input.

Definition at line 55 of file istream.

5.367.2 Member Typedef Documentation

5.367.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream< _CharT, _Traits >::__ctype_type`

These are non-standard types.

Reimplemented from [std::basic_ios](#)< [_CharT](#), [_Traits](#) >.

Definition at line 71 of file istream.

5.367.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits> >
std::basic_istream< _CharT, _Traits >::__num_get_type`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file `istream`.

5.367.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios< _CharT, _Traits >::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream< _CharT, _Traits >](#), [std::basic_ostream< char, _Traits >](#), and [std::basic_ostream< char >](#).

Definition at line 84 of file `basic_ios.h`.

5.367.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istream< _CharT, _Traits, _Alloc >](#).

Definition at line 59 of file `istream`.

5.367.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 444 of file [ios_base.h](#).

5.367.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file [ios_base.h](#).

5.367.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_istream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 60 of file `istream`.

5.367.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.367.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_istream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 62 of file `istream`.

5.367.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 369 of file *ios_base.h*.

5.367.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 61 of file *istream*.

5.367.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

_Ios_Seekdir is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK_CUR* in the C standard library.
- *end*, equivalent to *SEEK_END* in the C standard library.

Definition at line 401 of file *ios_base.h*.

5.367.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 63 of file `istream`.

5.367.3 Member Enumeration Documentation

5.367.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

5.367.4 Constructor & Destructor Documentation

5.367.4.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits>::basic_istream (__streambuf_type * __sb) [inline, explicit]`

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 91 of file `istream`.

5.367.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_istream< _CharT, _Traits >::~~basic_istream () [inline, virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 101 of file `istream`.

5.367.5 Member Function Documentation

5.367.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.367.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.367.5.3 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit) [inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file basic_ios.tcc.

Referenced by std::basic_ios< _CharT, _Traits >::rdbuf().

5.367.5.4 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of __rhs into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file basic_ios.tcc.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.367.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.367.5.6 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions (iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.367.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions () const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

5.367.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file basic_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.367.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill(char_type __ch) [inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.367.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.367.5.11 `fmtflags std::ios_base::flags() const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.367.5.12 `fmtflags std::ios_base::flags(fmtflags __fmtfl) [inline, inherited]`

Setting new format flags all at once.

Parameters

__fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 561 of file `ios_base.h`.

5.367.5.13 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount() const [inline]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.367.5.14 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (void)`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.15 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type & __c)`

Simple extraction.

Parameters

`c` The character in which to store data.

Returns

`*this`

Tries to extract a character and store it in `c`. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.16 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n, char_type __delim)`

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.367.5.17 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline]`

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file istream.

5.367.5.18 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb, char_type __delim)`

Extraction into another streambuf.

Parameters

- sb* A streambuf in which to store data.
delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::putc()`.

5.367.5.19 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::get (__streambuf_type &
__sb) [inline]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file `istream`.

5.367.5.20 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::getline (char_type * __s, streamsize __n, char_type __delim)`

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

delim A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.367.5.21 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline]

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file istream.

Referenced by std::basic_istream< char >::getline().

5.367.5.22 locale std::ios_base::getloc () const [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Traits >::imbue(), std::operator>>(), and std::ws().

5.367.5.23 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.367.5.24 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(void)`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sputc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.25 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(streamsize __n)`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.367.5.26 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(streamsize __n, int_type __delim)`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.367.5.27 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.367.5.28 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.367.5.29 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.367.5.30 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow(char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.367.5.31 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits>::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.367.5.32 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.367.5.33 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned int
& __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.367.5.34 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (__streambuf_type * __sb)`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.35 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (short & __n)`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.36 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

5.367.5.37 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long & __n
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.367.5.38 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long long &
__n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file istream.

5.367.5.39 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file istream.

5.367.5.40 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (bool & __n
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 167 of file istream.

5.367.5.41 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
long & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 189 of file istream.

5.367.5.42 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (ios_base
&(*)(ios_base &) __pf) [inline]`

Interface for manipulators.

Manipulators such as [std::ws](#) and [std::dec](#) use these functions in constructs like [std::cin >> std::ws](#). For more information, see the `iomanip` header.

Definition at line 131 of file istream.

5.367.5.43 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
long long & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 198 of file istream.

5.367.5.44 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
short & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 174 of file istream.

5.367.5.45 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (void *& __p
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

5.367.5.46 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (float & __f
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.367.5.47 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (double &
__f) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file istream.

5.367.5.48 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (int & __n)`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.367.5.49 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long double & __f) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

5.367.5.50 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek (void)`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.367.5.51 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.367.5.52 streamsize std::ios_base::precision () const [inline, inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

5.367.5.53 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (char_type __c)

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.367.5.54 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.367.5.55 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::rdbuf () const [inline, inherited]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits, _Alloc >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 311 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::imbue(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

5.367.5.56 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::clear().

5.367.5.57 `template<typename _CharT, typename _Traits> istate
std::basic_ios< _CharT, _Traits >::rdstate () const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.367.5.58 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
char_type * __s, streamsize __n)`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.59 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n)`

Extraction until the buffer is exhausted, but no more.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rddbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.60 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.367.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (pos_type __pos)`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.367.5.62 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (off_type __off, ios_base::seekdir __dir)`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.63 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.367.5.64 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.367.5.65 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.367.5.66 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.67 `static bool std::ios_base::sync_with_stdio (bool __sync = true)` **[static, inherited]**

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.367.5.68 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::pos_type std::basic_istream< _CharT, _Traits>::tellg (void)`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.367.5.69 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file basic_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.367.5.70 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file basic_ios.h.

5.367.5.71 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void)`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

5.367.5.72 void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.367.5.73 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline, inherited]

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.367.5.74 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.367.5.75 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.367.5.76 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.367.6 Member Data Documentation

5.367.6.1 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::_M_gcount` **[protected]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.367.6.2 `const fmtflags std::ios_base::adjustfield` **[static, inherited]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.367.6.3 `const openmode std::ios_base::app` **[static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.367.6.4 `const openmode std::ios_base::ate` **[static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.367.6.5 `const iostate std::ios_base::badbit` **[static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.367.6.6 `const fmtflags std::ios_base::basefield` **[static, inherited]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.367.6.7 `const seekdir std::ios_base::beg` **[static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

5.367.6.8 `const openmode std::ios_base::binary` **[static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.367.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.367.6.10 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.367.6.11 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

5.367.6.12 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.367.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

5.367.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.367.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.367.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.367.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.367.6.18 `const fmtflags std::ios_base::hex` [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.367.6.19 `const openmode std::ios_base::in` [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.367.6.20 `const fmtflags std::ios_base::internal` [static, inherited]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

5.367.6.21 `const fmtflags std::ios_base::left` [static, inherited]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.367.6.22 `const fmtflags std::ios_base::oct` [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.367.6.23 `const openmode std::ios_base::out` [static, inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file ios_base.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.367.6.24 `const fmtflags std::ios_base::right` `[static, inherited]`

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.367.6.25 `const fmtflags std::ios_base::scientific` `[static, inherited]`

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.367.6.26 `const fmtflags std::ios_base::showbase` `[static, inherited]`

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.367.6.27 `const fmtflags std::ios_base::showpoint` `[static, inherited]`

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios_base.h.

5.367.6.28 `const fmtflags std::ios_base::showpos` `[static, inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios_base.h.

5.367.6.29 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 308 of file ios_base.h.

5.367.6.30 `const openmode std::ios_base::trunc` [`static`, `inherited`]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.367.6.31 `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.367.6.32 `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

5.368 `std::basic_istream< _CharT, _Traits >::sentry` Class Reference

Performs setup work for input streams.

Public Types

- `typedef __istream_type::__ctype_type __ctype_type`
- `typedef _Traits::int_type __int_type`
- `typedef basic_istream< _CharT, _Traits > __istream_type`
- `typedef basic_streambuf< _CharT, _Traits > __streambuf_type`
- `typedef _Traits traits_type`

Public Member Functions

- `sentry (basic_istream< _CharT, _Traits > &__is, bool __noskipws=false)`
- `operator bool () const`

5.368.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _CharT, _Traits >::sentry
```

Performs setup work for input streams. Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 633 of file istream.

5.368.2 Member Typedef Documentation

5.368.2.1 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream< _CharT, _Traits >::sentry::traits_type`

Easy access to dependant types.

Definition at line 640 of file istream.

5.368.3 Constructor & Destructor Documentation

5.368.3.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::sentry (basic_istream< _CharT, _Traits > & __is, bool __noskipws = false) [explicit]`

The constructor performs all the work.

Parameters

is The input stream to guard.

noskipws Whether to consume whitespace or not.

If the stream state is good (*is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if *noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

5.369 `std::basic_istringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1807

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `istream.tcc`.

References `std::__ctype_abstract_base< _CharT >::is()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.368.4 Member Function Documentation

5.368.4.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::operator bool () const` [`inline`, `explicit`]

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 681 of file `istream`.

The documentation for this class was generated from the following files:

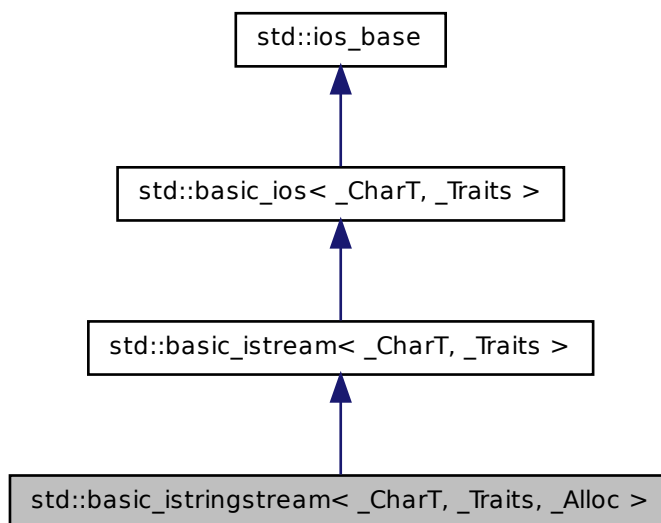
- [istream](#)
- [istream.tcc](#)

5.369 `std::basic_istringstream< _CharT, _Traits, _- Alloc >` Class Template Reference

Controlling input for `std::string`.

This class supports reading from objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_istream< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`

- typedef _Ios_Iostate [iostate](#)
 - typedef traits_type::off_type [off_type](#)
 - typedef int [open_mode](#)
 - typedef _Ios_Openmode [openmode](#)
 - typedef traits_type::pos_type [pos_type](#)
 - typedef int [seek_dir](#)
 - typedef _Ios_Seekdir [seekdir](#)
 - typedef [std::streamoff](#) [streamoff](#)
 - typedef [std::streampos](#) [streampos](#)
 - typedef _Traits [traits_type](#)
-
- typedef [num_put](#)< _CharT, [ostreambuf_iterator](#)< _CharT, _Traits > > [__num_put_type](#)

Public Member Functions

- [basic_istringstream](#) ([ios_base::openmode](#) __mode=[ios_base::in](#))
- [basic_istringstream](#) (const [__string_type](#) &__str, [ios_base::openmode](#) __mode=[ios_base::in](#))
- [~basic_istringstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [streamsize](#) [gcount](#) () const
- template<>
[basic_istream](#)< [char](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- template<>
[basic_istream](#)< [wchar_t](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const

- `template<>`
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
 - `template<>`
`basic_istream< char > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
 - `locale imbue (const locale &__loc)`
 - `long & iword (int __ix)`
 - `char narrow (char_type __c, char __dfault) const`
 - `streamsize precision () const`
 - `streamsize precision (streamsize __prec)`
 - `void *& pword (int __ix)`
 - `__stringbuf_type * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iosstate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `void setstate (iosstate __state)`
 - `void str (const __string_type &__s)`
 - `__string_type str () const`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width (streamsize __wide)`
 - `streamsize width () const`
-
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
 - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
 - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`

- [streamsize](#) [readsome](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [putback](#) ([char_type](#) __c)
- [__istream_type](#) & [unget](#) ()
- [int](#) [sync](#) ()
- [pos_type](#) [tellg](#) ()
- [__istream_type](#) & [seekg](#) ([pos_type](#))
- [__istream_type](#) & [seekg](#) ([off_type](#), [ios_base::seekdir](#))
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static [bool](#) [sync_with_stdio](#) ([bool](#) __sync=true)
- static [int](#) [xalloc](#) () [throw](#) ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)

- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- void `_M_cache_locale` (const `locale` &__loc)
- void `_M_call_callbacks` (`event` __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename _ValueT >
 `__istream_type` & `_M_extract` (_ValueT &__v)
- `_Words` & `_M_grow_words` (int __index, bool __iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`<_CharT, _Traits> *__sb)

Protected Attributes

- `_Callback_list` * `_M_callbacks`
- const `__ctype_type` * `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` * `_M_num_get`
- const `__num_put_type` * `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`<_CharT, _Traits> * `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`<_CharT, _Traits> * `_M_tie`
- `streamsize` `_M_width`
- `_Words` * `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

Friends

- class `sentry`

5.369.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc> class `std::basic_istream<_CharT, _Traits, _Alloc>`

Controlling input for `std::string`.

This class supports reading from objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as `sb`.

Definition at line 256 of file `sstream`.

5.369.2 Member Typedef Documentation

5.369.2.1 `template<typename _CharT, typename _Traits> typedef c_type<_CharT> std::basic_istream<_CharT, _Traits>::__c_type_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `istream`.

5.369.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_istream<_CharT, _Traits>::__num_get_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file `istream`.

5.369.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]`

These are non-standard types.

5.369 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1815

Reimplemented in [std::basic_ostream< _CharT, _Traits >](#), [std::basic_ostream< char, _Traits >](#), and [std::basic_ostream< char >](#).

Definition at line 84 of file `basic_ios.h`.

5.369.2.4 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _CharT std::basic_istream< _CharT, _Traits, _Alloc
>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 260 of file `sstream`.

5.369.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.369.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`

- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file ios_base.h.

5.369.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
 typedef traits_type::int_type std::basic_istream< _CharT,
 _Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 265 of file sstream.

5.369.2.8 `typedef _Ios_Iostate std::ios_base::iostate` **[inherited]**

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1817

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.369.2.9 `template<typename _CharT, typename _Traits, typename _Alloc>`
`typedef traits_type::off_type std::basic_istream<_CharT,`
`_Traits, _Alloc>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream<_CharT, _Traits>](#).

Definition at line 267 of file `sstream`.

5.369.2.10 `typedef _Ios_Openmode std::ios_base::openmode` `[inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

5.369.2.11 `template<typename _CharT, typename _Traits, typename _Alloc>
 typedef traits_type::pos_type std::basic_istream< _CharT,
 _Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 266 of file `sstream`.

5.369.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

5.369.2.13 `template<typename _CharT, typename _Traits, typename _Alloc>
 typedef _Traits std::basic_istream< _CharT, _Traits, _Alloc
 >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 261 of file `sstream`.

5.369.3 Member Enumeration Documentation

5.369.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

5.369.4 Constructor & Destructor Documentation

5.369.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_istream< _CharT, _Traits, _Alloc
>::basic_istream (ios_base::openmode __mode = ios_base::in
) [inline, explicit]`

Default constructor starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using `mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 292 of file `sstream`.

5.369.4.2 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_istream< _CharT, _Traits, _Alloc
>::basic_istream (const __string_type & __str,
ios_base::openmode __mode = ios_base::in) [inline,
explicit]`

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using *str* and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 310 of file `sstream`.

5.369.4.3 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_istream< _CharT, _Traits, _Alloc
>::~~basic_istream () [inline]`

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 321 of file sstream.

5.369.5 Member Function Documentation

5.369.5.1 `const locale& std::ios_base::_M_getloc () const [inline,
inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.369.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1821

5.369.5.3 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.369.5.4 `template<typename _CharT, typename _Traits> basic_ios<
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that [rdbuf\(\)](#) and [rdstate\(\)](#) remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.369.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::eof () const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.369.5.6 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.369.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference

1823

```
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.369.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const` `[inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::regex_traits<_Ch_type>::value()`.

5.369.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill() const` `[inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.369.5.10 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.369.5.11 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.369.5.12 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1825

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.369.5.13 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::gcount () const
[inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.369.5.14 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.15 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.16 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.369.5.17 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

5.369.5.18 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n, char_type __delim)
[inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- $n-1$ characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.369.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb, widen('\n'))`.

Definition at line 366 of file istream.

5.369.5.20 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]`

String extraction.

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1829

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

`*this`

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sputc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.369.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

5.369.5.22 locale std::ios_base::getloc () const [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

5.369.5.23 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good () const [inline, inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.369.5.24 template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::ignore (streamsize __n) [inherited]

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.369.5.25 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.26 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.369.5.27 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc) [inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference **1833**

5.369.5.28 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.369.5.29 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.369.5.30 `template<typename _CharT, typename _Traits> char
std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.369.5.31 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const` **[inline, inherited]**

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

5.369.5.32 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! () const` **[inline, inherited]**

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

5.369.5.33 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> (short & __n)` **[inherited]**

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1835

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

5.369.5.34 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.369.5.35 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

5.369.5.36 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 174 of file istream.

```
5.369.5.37  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_istream< _CharT, _Traits >::operator>> ( void *& __p
             ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

```
5.369.5.38  template<typename _CharT , typename _Traits > basic_istream<
             _CharT, _Traits > & std::basic_istream< _CharT, _Traits
             >::operator>> ( int & __n ) [inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

5.369 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1837

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.39 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.40 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.369.5.41 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

5.369.5.42 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

5.369.5.43 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (ios_base
&(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file `istream`.

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1839

5.369.5.44 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.369.5.45 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file istream.

5.369.5.46 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

5.369.5.47 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long long &
__n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.369.5.48 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1841

5.369.5.49 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

5.369.5.50 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.51 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits >::copyfmt(), and std::operator<<().

5.369.5.52 `streamsize std::ios_base::precision (streamsize __prec)`
[inline, inherited]

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.369.5.53 `template<typename _CharT, typename _Traits > basic_istream<`
`_CharT, _Traits > & std::basic_istream< _CharT, _Traits`
`>::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream<_CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios<_CharT, _Traits >::eof(), std::ios_base::goodbit, std::basic_ios<

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1843

`_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.369.5.54 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.369.5.55 `template<typename _CharT, typename _Traits, typename _Alloc> __stringbuf_type* std::basic_istream<_CharT, _Traits, _Alloc>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 332 of file `sstream`.

5.369.5.56 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.369.5.57 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.369.5.58 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::read(
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.59 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.60 `void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.369.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

`off` A file offset object.

`dir` The direction in which to seek.

Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1847

Definition at line 870 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.369.5.62 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.369.5.63 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.369.5.64 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.369.5.65 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate __state) [inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream<`

5.369 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1849

_CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

5.369.5.66 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istream< _CharT, _Traits, _Alloc >::str (
const __string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Calls `rdbuf() -> str(s)`.

Definition at line 350 of file sstream.

5.369.5.67 `template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istream< _CharT, _Traits, _Alloc
>::str () const [inline]`

Copying out the string buffer.

Returns

`rdbuf() -> str()`

Definition at line 340 of file sstream.

5.369.5.68 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.69 `static bool std::ios_base::sync_with_stdio (bool __sync = true)` **[static, inherited]**

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.369.5.70 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (void)` **[inherited]**

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1851

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.369.5.71 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.369.5.72 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

5.369.5.73 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void) [inherited]`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

5.369.5.74 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.369.5.75 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen (char __c) const [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1853

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.369.5.76 `streamsize std::ios_base::width () const` [`inline`, `inherited`]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, and `std::operator>>()`.

5.369.5.77 `streamsize std::ios_base::width (streamsize __wide)` [`inline`, `inherited`]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.369.5.78 `static int std::ios_base::xalloc () throw ()` [`static`, `inherited`]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iwor`d and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iwor`d and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iwor`d and `pword` arrays.

5.369.6 Member Data Documentation
**5.369.6.1 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::_M_gcount` [protected,
inherited]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsomewhat()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.369.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.369.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.369.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

5.369 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference 1855

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.369.6.5 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.369.6.6 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.369.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.369.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.369.6.9 `const fmtflags std::ios_base::boolalpha` **[static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.369.6.10 `const seekdir std::ios_base::cur` **[static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.369.6.11 `const fmtflags std::ios_base::dec` **[static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

5.369.6.12 `const seekdir std::ios_base::end` **[static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.369.6.13 `const iostate std::ios_base::eofbit` **[static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits`

5.369 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference 1857

>::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.369.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.369.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.369.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of set f.

Definition at line 324 of file ios_base.h.

5.369.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(),

`std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsomewhat()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.369.6.18 `const fmtflags std::ios_base::hex [static, inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.369.6.19 `const openmode std::ios_base::in [static, inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.369.6.20 `const fmtflags std::ios_base::internal [static, inherited]`

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

5.369.6.21 `const fmtflags std::ios_base::left [static, inherited]`

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.369.6.22 `const fmtflags std::ios_base::oct [static, inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

5.369 `std::basic_istringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1859

Referenced by `std::basic_ostream<_CharT, _Traits>::operator<<()`.

5.369.6.23 `const openmode std::ios_base::out` [`static`, `inherited`]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

5.369.6.24 `const fmtflags std::ios_base::right` [`static`, `inherited`]

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file `ios_base.h`.

5.369.6.25 `const fmtflags std::ios_base::scientific` [`static`, `inherited`]

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

5.369.6.26 `const fmtflags std::ios_base::showbase` [`static`, `inherited`]

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file `ios_base.h`.

5.369.6.27 `const fmtflags std::ios_base::showpoint` [`static`, `inherited`]

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 302 of file `ios_base.h`.

5.369.6.28 `const fmtflags std::ios_base::showpos` [`static`, `inherited`]

Generates a `+` sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

5.369.6.29 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.369.6.30 `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.369.6.31 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.369.6.32 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

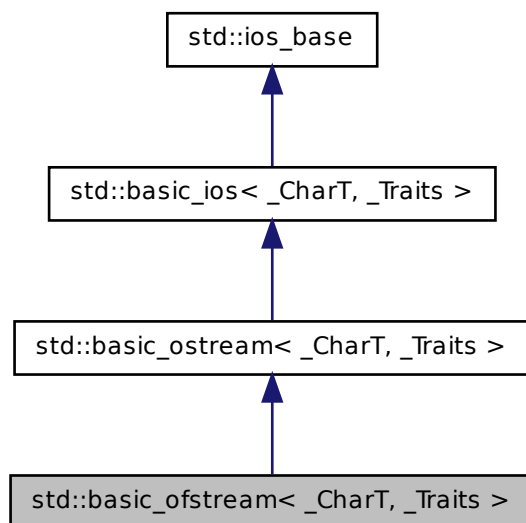
- [sstream](#)

5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference

Controlling output for files.

This class supports reading from named files, using the inherited functions from [std::basic_ostream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for std::basic_ofstream< _CharT, _Traits >:



Public Types

- typedef [ctype](#)< _CharT > [__ctype_type](#)
- typedef [basic_filebuf](#)< [char_type](#), [traits_type](#) > [__filebuf_type](#)
- typedef [basic_ios](#)< _CharT, _Traits > [__ios_type](#)
- typedef [num_put](#)< _CharT, [ostreambuf_iterator](#)< _CharT, _Traits > > [__num_put_type](#)
- typedef [basic_ostream](#)< [char_type](#), [traits_type](#) > [__ostream_type](#)
- typedef [basic_streambuf](#)< _CharT, _Traits > [__streambuf_type](#)
- typedef _CharT [char_type](#)
- enum [event](#) { [erase_event](#), [imbue_event](#), [copyfmt_event](#) }
- typedef void(* [event_callback](#))([event](#), [ios_base](#) &, int)
- typedef _Ios_Fmtflags [fmtflags](#)
- typedef traits_type::int_type [int_type](#)
- typedef int [io_state](#)
- typedef _Ios_Iostate [iostate](#)
- typedef traits_type::off_type [off_type](#)

- typedef int **open_mode**
 - typedef _Ios_Openmode [openmode](#)
 - typedef traits_type::pos_type [pos_type](#)
 - typedef int **seek_dir**
 - typedef _Ios_Seekdir [seekdir](#)
 - typedef [std::streamoff](#) **streamoff**
 - typedef [std::streampos](#) **streampos**
 - typedef _Traits [traits_type](#)
-
- typedef [num_get](#)< _CharT, [istreambuf_iterator](#)< _CharT, _Traits > > [__num_get_type](#)

Public Member Functions

- [basic_ofstream](#) ()
- [basic_ofstream](#) (const char *__s, [ios_base::openmode](#) __mode=[ios_base::out|ios_base::trunc](#))
- [basic_ofstream](#) (const [std::string](#) &__s, [ios_base::openmode](#) __mode=[ios_base::out|ios_base::trunc](#))
- [~basic_ofstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- void [close](#) ()
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [__ostream_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- bool [is_open](#) ()
- bool [is_open](#) () const
- long & [iword](#) (int __ix)
- char [narrow](#) ([char_type](#) __c, char __dfault) const

- `void open (const char *__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
- `void open (const std::string &__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `__filebuf_type * rdbuf () const`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> *__sb)`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `__ostream_type & seekp (pos_type)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `pos_type tellp ()`
- `basic_ostream<_CharT, _Traits> * tie () const`
- `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __t, __tstr)`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width (streamsize __wide)`
- `streamsize width () const`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ofstream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`

- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`

- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & [_M_insert](#) (_ValueT __v)
- void [init](#) ([basic_streambuf](#)<_CharT, _Traits> * __sb)

Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

Friends

- `class sentry`

5.370.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ofstream< _CharT, _Traits >`

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 583 of file `fstream`.

5.370.2 Member Typedef Documentation

5.370.2.1 `template<typename _CharT, typename _Traits> typedef
 ctype<_CharT> std::basic_ofstream<_CharT, _Traits
 >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `ostream`.

5.370.2.2 `template<typename _CharT, typename _Traits> typedef
 num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
 > std::basic_ios<_CharT, _Traits>::__num_get_type
 [inherited]`

These are non-standard types.

Reimplemented in [std::basic_istream<_CharT, _Traits>](#), [std::basic_istream<char, _Traits>](#), and [std::basic_istream<char>](#).

Definition at line 86 of file `basic_ios.h`.

5.370.2.3 `template<typename _CharT, typename _Traits> typedef
 num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
 > std::basic_ofstream<_CharT, _Traits>::__num_put_type
 [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file `ostream`.

5.370.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
 std::basic_ofstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ofstream<_CharT, _Traits>](#).

Definition at line 587 of file `fstream`.

5.370.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`
[inherited]

The type of an event callback function.

Parameters

- event* One of the members of the event enum.
- ios_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.370.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.370.2.7 `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_ofstream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Definition at line 589 of file `fstream`.

5.370.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.370.2.9 `template<typename _CharT , typename _Traits > typedef traits_type::off_type std::basic_ofstream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 591 of file fstream.

5.370.2.10 **typedef _Ios_Openmode std::ios_base::openmode [inherited]**

This is a bitmask type.

_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 369 of file ios_base.h.

5.370.2.11 **template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_ofstream< _CharT, _Traits >::pos_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 590 of file fstream.

5.370.2.12 **typedef _Ios_Seekdir std::ios_base::seekdir [inherited]**

This is an enumerated type.

_Ios_Seekdir is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK_CUR* in the C standard library.
- *end*, equivalent to *SEEK_END* in the C standard library.

Definition at line 401 of file ios_base.h.

5.370.2.13 `template<typename _CharT , typename _Traits > typedef _Traits
std::basic_ofstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 588 of file fstream.

5.370.3 Member Enumeration Documentation

5.370.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 427 of file ios_base.h.

5.370.4 Constructor & Destructor Documentation

5.370.4.1 `template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream ()
[inline]`

Default constructor.

Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 609 of file fstream.

5.370.4.2 `template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (const char
* __s, ios_base::openmode __mode = ios_base::out|ios_base::trunc)
[inline, explicit]`

Create an output file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::out](#)|[ios_base::trunc](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 624 of file `fstream`.

```
5.370.4.3 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream
( const std::string & __s, ios_base::openmode __mode =
ios_base::out|ios_base::trunc ) [inline, explicit]
```

Create an output file stream.

Parameters

s `std::string` specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::out](#)|[ios_base::trunc](#) is automatically included in *mode*.

Definition at line 642 of file `fstream`.

```
5.370.4.4 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream ( )
[inline]
```

The destructor does nothing.

The file is closed by the `filebuf` object, not the formatting stream.

Definition at line 657 of file `fstream`.

5.370.5 Member Function Documentation

```
5.370.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline,
inherited]
```

Locale access.

Returns

A reference to the current locale.

5.370 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1873

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

5.370.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream< _CharT, _Traits >::M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.370.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.370.5.4 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.370.5.5 `template<typename _CharT, typename _Traits> void
std::basic_ofstream<_CharT, _Traits>::close () [inline]`

Close the file.

Calls [std::basic_filebuf::close\(\)](#). If that function fails, `failbit` is set in the stream's error state.

Definition at line 737 of file `fstream`.

5.370.5.6 `template<typename _CharT, typename _Traits> basic_ios<
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that [rdbuf\(\)](#) and [rdstate\(\)](#) remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

5.370 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1875

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), and std::ios_base::width().

5.370.5.7 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof() const [inline, inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), and std::basic_istream< _CharT, _Traits >::unget().

5.370.5.8 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions() const [inline, inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.370.5.9 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions(iostate __except) [inline, inherited]

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.370.5.10 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::fail () const [inline,
inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.370.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.370.5.12 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill () const [inline,
inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.370.5.13 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.370.5.14 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.370.5.15 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.370.5.16 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.370.5.17 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 170 of file basic_ios.h.

5.370.5.18 `template<typename _CharT, typename _Traits> locale
std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.370.5.19 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.370.5.20 `template<typename _CharT, typename _Traits> bool
std::basic_ofstream<_CharT, _Traits>::is_open() [inline]`

Wrapper to test for an open file.

Returns

`rdbuf() -> is_open()`

Definition at line 676 of file fstream.

5.370.5.21 `long& std::ios_base::iword(int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.370.5.22 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow(char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

5.370 std::basic_ofstream<_CharT, _Traits> Class Template Reference 1881

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

```
5.370.5.23 template<typename _CharT , typename _Traits > void  
std::basic_ofstream<_CharT, _Traits >::open ( const char * __s,  
ios_base::openmode __mode = ios_base::out | ios_base::trunc )  
[inline]
```

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 697 of file `fstream`.

```
5.370.5.24 template<typename _CharT , typename _Traits > void  
std::basic_ofstream<_CharT, _Traits >::open ( const std::string &  
__s, ios_base::openmode __mode = ios_base::out | ios_base::trunc  
) [inline]
```

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 718 of file `fstream`.

5.370.5.25 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.370.5.26 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.370.5.27 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

5.370.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

5.370 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1883

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

```
5.370.5.29  template<typename _CharT, typename _Traits > basic_ofstream<
               _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits
               >::operator<<( __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

```
5.370.5.30  template<typename _CharT, typename _Traits> __ostream_type&
               std::basic_ofstream< _CharT, _Traits >::operator<<( ios_base
               &(*)(ios_base &) __pf ) [inline, inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file ostream.

5.370.5.31 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.370.5.32 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.370.5.33 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.370.5.34 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.370.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<< (long double __f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.370.5.36 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.370.5.37 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.370.5.38 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.370.5.39 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ofstream<_CharT, _Traits>::operator<<(bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.370.5.40 `template<typename _CharT, typename _Traits> basic_ofstream<
_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits
>::operator<<(short __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.370.5.41 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.370.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.370.5.43 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1889

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

5.370.5.44 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.370.5.45 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

5.370.5.46 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.370.5.47 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.370.5.48 `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_ofstream<_CharT, _Traits>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current [basic_filebuf](#) buffer.

This hides both signatures of [std::basic_ios::rdbuf\(\)](#).

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 668 of file fstream.

5.370.5.49 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.370.5.50 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate () const [inline, inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.370.5.51 `void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.370.5.52 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

- off* A file offset object.
- dir* The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.53 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (pos_type __pos) [inherited]`

Changing the current write position.

Parameters

- pos* A file position object.

Returns

*this

5.370 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1893

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.54 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file ios_base.h.

5.370.5.55 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.370.5.56 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.370.5.57 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.370.5.58 `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>::pos_type std::basic_ofstream<_CharT, _Traits>::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.370.5.59 `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ofstream<_CharT, _Traits>* __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

5.370.5.60 `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.370.5.61 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.370.5.62 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen (char __c) const [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.370.5.63 streamsize std::ios_base::width (streamsize __*wide*) [inline, inherited]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file ios_base.h.

5.370.5.64 streamsize std::ios_base::width () const [inline, inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

5.370.5.65 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::write (const char_type * __*s*, streamsize __*n*) [inherited]

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- n characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.370.5.66 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.370.6 Member Data Documentation

5.370.6.1 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.370.6.2 `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.370.6.3 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.370.6.4 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.370.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.370.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.370.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.370.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.370.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.370.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.370.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.370.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.370.6.13 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.370.6.14 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.370.6.15 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.370.6.16 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

`std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.370.6.17 `const fmtflags std::ios_base::hex` [**static, inherited**]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.370.6.18 `const openmode std::ios_base::in` [**static, inherited**]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.370.6.19 `const fmtflags std::ios_base::internal` [**static, inherited**]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

5.370.6.20 `const fmtflags std::ios_base::left` [**static, inherited**]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.370.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.370.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.370.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.370.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.370.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.370.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios_base.h.

5.370.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

5.370.6.28 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.370.6.29 `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.370.6.30 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.370.6.31 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

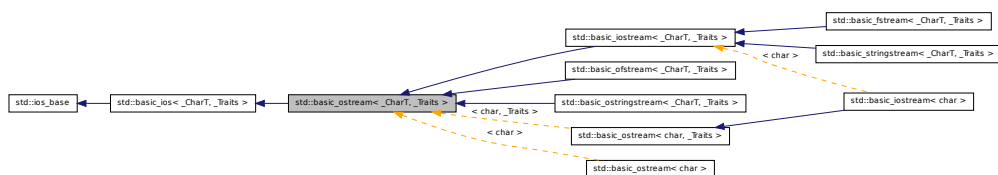
- [fstream](#)

5.371 `std::basic_ostream<_CharT, _Traits>` Class Template Reference

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual output.

Inheritance diagram for `std::basic_ostream<_CharT, _Traits>`:



Classes

- class `sentry`
Performs setup work for output streams.

Public Types

- typedef `ctype`< `_CharT` > `__ctype_type`
 - typedef `basic_ios`< `_CharT`, `_Traits` > `__ios_type`
 - typedef `num_put`< `_CharT`, `ostreambuf_iterator`< `_CharT`, `_Traits` > > `__num_put_type`
 - typedef `basic_ostream`< `_CharT`, `_Traits` > `__ostream_type`
 - typedef `basic_streambuf`< `_CharT`, `_Traits` > `__streambuf_type`
 - typedef `_CharT` `char_type`
 - enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
 - typedef void(* `event_callback`)(`event`, `ios_base` &, int)
 - typedef `_Ios_Fmtflags` `fmtflags`
 - typedef `_Traits::int_type` `int_type`
 - typedef int `io_state`
 - typedef `_Ios_Iostate` `iostate`
 - typedef `_Traits::off_type` `off_type`
 - typedef int `open_mode`
 - typedef `_Ios_Openmode` `openmode`
 - typedef `_Traits::pos_type` `pos_type`
 - typedef int `seek_dir`
 - typedef `_Ios_Seekdir` `seekdir`
 - typedef `std::streamoff` `streamoff`
 - typedef `std::streampos` `streampos`
 - typedef `_Traits` `traits_type`
-
- typedef `num_get`< `_CharT`, `istreambuf_iterator`< `_CharT`, `_Traits` > > `__num_get_type`

Public Member Functions

- [basic_ostream](#) ([__streambuf_type](#) *__sb)
- virtual [~basic_ostream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- void [exceptions](#) ([iostate](#) __except)
- [iostate](#) [exceptions](#) () const
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [__ostream_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- long & [iword](#) (int __ix)
- char [narrow](#) ([char_type](#) __c, char __dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- void *& [pword](#) (int __ix)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) () const
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *__sb)
- [iostate](#) [rdstate](#) () const
- void [register_callback](#) ([event_callback](#) __fn, int __index)
- [__ostream_type](#) & [seekp](#) ([pos_type](#))
- [__ostream_type](#) & [seekp](#) ([off_type](#), [ios_base::seekdir](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- void [setstate](#) ([iostate](#) __state)
- [pos_type](#) [tellp](#) ()
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) ([basic_ostream](#)< [_CharT](#), [_Traits](#) > *__tiestr)
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- void [unsetf](#) ([fmtflags](#) __mask)
- [char_type](#) [widen](#) (char __c) const
- [streamsize](#) [width](#) () const

- `streamsize width (streamsize __wide)`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`

- void [_M_write](#) (const [char_type](#) *__s, [streamsize](#) __n)
- [__ostream_type](#) & [write](#) (const [char_type](#) *__s, [streamsize](#) __n)
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static [bool sync_with_stdio](#) (bool __sync=true)
- static [int xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags adjustfield](#)
- static const [openmode app](#)
- static const [openmode ate](#)
- static const [iostate badbit](#)
- static const [fmtflags basefield](#)
- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const [locale](#) &__loc)
- void **_M_call_callbacks** ([event](#) __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- [_Words](#) & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & **_M_insert** (_ValueT __v)
- void **init** ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * **_M_callbacks**
- const [__ctype_type](#) * **_M_ctype**
- [iostate](#) **_M_exception**
- [char_type](#) **_M_fill**
- bool **_M_fill_init**
- [fmtflags](#) **_M_flags**
- [locale](#) **_M_ios_locale**
- [_Words](#) **_M_local_word** [**_S_local_word_size**]
- const [__num_get_type](#) * **_M_num_get**
- const [__num_put_type](#) * **_M_num_put**
- [streamsize](#) **_M_precision**
- [basic_streambuf](#)< _CharT, _Traits > * **_M_streambuf**
- [iostate](#) **_M_streambuf_state**
- [basic_ostream](#)< _CharT, _Traits > * **_M_tie**
- [streamsize](#) **_M_width**
- [_Words](#) * **_M_word**
- int **_M_word_size**
- [_Words](#) **_M_word_zero**

Friends

- class **sentry**

5.371.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ostream< _-  
CharT, _Traits >
```

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual output.

Definition at line 55 of file ostream.

5.371.2 Member Typedef Documentation

5.371.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ostream< _CharT, _Traits
>::__ctype_type`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file ostream.

5.371.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_ios< _CharT, _Traits >::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_istream< _CharT, _Traits >](#), [std::basic_istream< char, _Traits >](#), and [std::basic_istream< char >](#).

Definition at line 86 of file basic_ios.h.

5.371.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> >
std::basic_ostream< _CharT, _Traits >::__num_put_type`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file ostream.

5.371.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ostream< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_iostream< _CharT, _Traits >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 59 of file `ostream`.

5.371.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the [ios_base](#) object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 444 of file `ios_base.h`.

5.371.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`

- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.371.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ostream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_iostream< _CharT, _Traits >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_iostream< char >](#).

Definition at line 60 of file `ostream`.

5.371.2.8 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.371.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ostream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_iostream< _CharT, _Traits >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 62 of file `ostream`.

5.371.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`

- out
- trunc

Definition at line 369 of file ios_base.h.

5.371.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ostream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 61 of file ostream.

5.371.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file ios_base.h.

5.371.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ostream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream<`

[_CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 63 of file ostream.

5.371.3 Member Enumeration Documentation

5.371.3.1 enum std::ios_base::event [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 427 of file ios_base.h.

5.371.4 Constructor & Destructor Documentation

5.371.4.1 template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::basic_ostream (__streambuf_type * __sb) [inline, explicit]

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 82 of file ostream.

5.371.4.2 template<typename _CharT, typename _Traits> virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream () [inline, virtual]

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 91 of file ostream.

5.371.5 Member Function Documentation

5.371.5.1 const locale& std::ios_base::_M_getloc () const [inline, inherited]

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

5.371.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream<_CharT, _Traits>::_M_write (const char_type
* __s, streamsize __n) [inline]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

5.371.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.371.5.4 `template<typename _CharT, typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.371.5.5 `template<typename _CharT, typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.371.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::eof () const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.371.5.7 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.371.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

5.371 std::basic_ostream< _CharT, _Traits > Class Template Reference 1919

```
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file basic_ios.h.

5.371.5.9 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail() const [inline, inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file basic_ios.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

5.371.5.10 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const [inline, inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.371.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.371.5.12 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.371.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

5.371.5.14 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

5.371.5.15 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

5.371.5.16 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.371.5.17 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.371.5.18 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.371.5.19 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.371.5.20 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.371.5.21 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.371.5.22 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.371.5.23 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iosmanip` header.

Definition at line 117 of file `ostream`.

5.371.5.24 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.371.5.25 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb)`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.371.5.26 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.371.5.27 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n)`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.371.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(__ostream_type &(*)(__ostream_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file ostream.

5.371.5.29 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(unsigned int __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file `ostream`.

5.371.5.30 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file `ostream`.

5.371.5.31 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file `ostream`.

5.371.5.32 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.371.5.33 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.371.5.34 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.371.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (double __f) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.371.5.36 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (ios_base &(*)(ios_base &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `io manip` header.

Definition at line 127 of file ostream.

5.371.5.37 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (short __n)`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References [std::ios_base::basefield](#), [std::ios_base::flags\(\)](#), [std::ios_base::hex](#), and [std::ios_base::oct](#).

5.371.5.38 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (float __f)
[inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.371.5.39 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (long double
__f) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.371.5.40 `streamsize std::ios_base::precision (streamsize __prec)`
`[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

5.371.5.41 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

5.371.5.42 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (char_type __c)`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.371.5.43 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.371.5.44 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

`sb` The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

5.371 std::basic_ostream< _CharT, _Traits > Class Template Reference 1933

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::clear().

5.371.5.45 `template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,
_Traits>::rdbuf() const [inline, inherited]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_istream< _CharT, _Traits, _Alloc >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_stringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 311 of file basic_ios.h.

Referenced by [std::basic_ostream< _CharT, _Traits >::flush\(\)](#), [std::basic_istream< _CharT, _Traits >::get\(\)](#), [std::basic_istream< _CharT, _Traits >::getline\(\)](#), [std::getline\(\)](#), [std::basic_istream< _CharT, _Traits >::ignore\(\)](#), [std::basic_ios< _CharT, _Traits >::imbue\(\)](#), [std::basic_ostream< _CharT, _Traits >::operator<<\(\)](#), [std::basic_istream< _CharT, _Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic_istream< _CharT, _Traits >::peek\(\)](#), [std::basic_ostream< _CharT, _Traits >::put\(\)](#), [std::basic_istream< _CharT, _Traits >::putback\(\)](#), [std::basic_istream< _CharT, _Traits >::read\(\)](#), [std::basic_istream< _CharT, _Traits >::readsome\(\)](#), [std::basic_istream< _CharT, _Traits >::seekg\(\)](#), [std::basic_ostream< _CharT, _Traits >::seekp\(\)](#), [std::basic_istream< _CharT, _Traits >::sync\(\)](#), [std::basic_istream< _CharT, _Traits >::tellg\(\)](#), [std::basic_ostream< _CharT, _Traits >::tellp\(\)](#), [std::basic_istream< _CharT, _Traits >::unget\(\)](#), and [std::ws\(\)](#).

5.371.5.46 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.371.5.47 `void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.371.5.48 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & std::basic_ostream< _CharT, _Traits>::seekp (pos_type __pos)`

Changing the current write position.

Parameters

`pos` A file position object.

Returns

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

5.371.5.49 `template<typename _CharT, typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp
(off_type __off, ios_base::seekdir __dir)`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.371.5.50 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask)
[inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file ios_base.h.

5.371.5.51 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.371.5.52 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic_ios.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.371.5.53 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.371.5.54 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>::pos_type std::basic_ostream< _CharT, _Traits>::tellp ()`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits>::rdbuf()`.

5.371.5.55 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>* std::basic_ios< _CharT, _Traits>::tie () const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`.

5.371.5.56 `template<typename _CharT, typename _Traits>
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

5.371.5.57 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.371.5.58 `template<typename _CharT, typename _Traits> char_type
 std::basic_ios<_CharT, _Traits>::widen (char __c) const
 [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

5.371 std::basic_ostream<_CharT, _Traits> Class Template Reference 1939

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.371.5.59 streamsize std::ios_base::width () const [inline, inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutTiter>::do_put()`, and `std::operator>>()`.

5.371.5.60 streamsize std::ios_base::width (streamsize __wide) [inline, inherited]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.371.5.61 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n)

Character string insertion.

Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.371.5.62 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.371.6 Member Data Documentation

5.371.6.1 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.371.6.2 const openmode std::ios_base::app [static, inherited]

Seek to end before each write.

Definition at line 372 of file ios_base.h.

5.371.6.3 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.371.6.4 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.371.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.371.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.371.6.7 `const openmode std::ios_base::binary` `[static, inherited]`

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file ios_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.371.6.8 `const fmtflags std::ios_base::boolalpha` `[static, inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.371.6.9 `const seekdir std::ios_base::cur` `[static, inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.371.6.10 `const fmtflags std::ios_base::dec` `[static, inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.371.6.11 `const seekdir std::ios_base::end` `[static, inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.371.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.371.6.13 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.371.6.14 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.371.6.15 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.371.6.16 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.371.6.17 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.371.6.18 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.371.6.19 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios_base.h.

5.371.6.20 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.371.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.371.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.371.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.371.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.371.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.371.6.26 `const fmtflags std::ios_base::showpoint` [`static, inherited`]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 302 of file `ios_base.h`.

5.371.6.27 `const fmtflags std::ios_base::showpos` [`static, inherited`]

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

5.371.6.28 `const fmtflags std::ios_base::skipws` [`static, inherited`]

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.371.6.29 `const openmode std::ios_base::trunc` [`static, inherited`]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.371.6.30 `const fmtflags std::ios_base::unitbuf` [`static, inherited`]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.371.6.31 `const fmtflags std::ios_base::uppercase` [`static, inherited`]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

5.372 `std::basic_ostream< _CharT, _Traits >::sentry` Class Reference

Performs setup work for output streams.

Public Member Functions

- `sentry` (`basic_ostream< _CharT, _Traits > &__os`)
- `~sentry` ()
- `operator bool` () const

5.372.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ostream< _CharT, _Traits >::sentry`

Performs setup work for output streams. Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 377 of file ostream.

5.372.2 Constructor & Destructor Documentation

5.372.2.1 `template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
basic_ostream< _CharT, _Traits > & __os) [explicit]`

The constructor performs preparatory work.

Parameters

os The output stream to guard.

If the stream state is good (*os.good()* is true), then if the stream is tied to another output stream, `is.tie()->flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file ostream.tcc.

5.372.2.2 `template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::~sentry ()
[inline]`

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 405 of file `ostream`.

References `std::uncaught_exception()`.

5.372.3 Member Function Documentation

5.372.3.1 `template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::operator bool ()
const [inline, explicit]`

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 426 of file `ostream`.

The documentation for this class was generated from the following files:

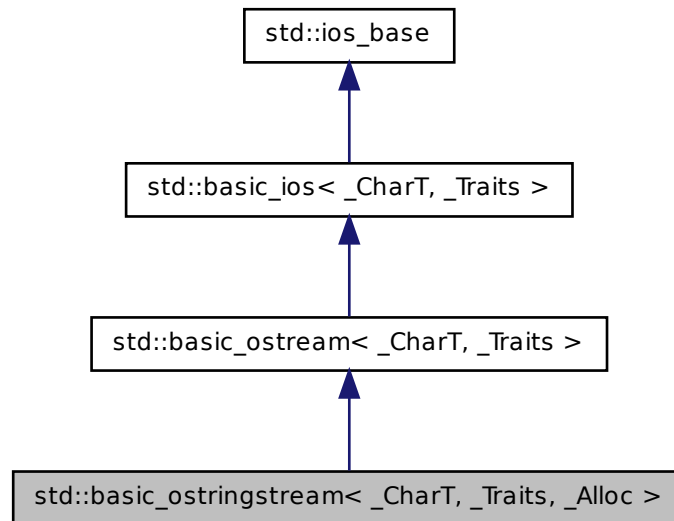
- [ostream](#)
- [ostream.tcc](#)

5.373 `std::basic_ostringstream< _CharT, _Traits, _- Alloc >` Class Template Reference

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ostringstream< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`

- typedef `_Ios_Iostate` [iostate](#)
 - typedef `traits_type::off_type` [off_type](#)
 - typedef `int` **`open_mode`**
 - typedef `_Ios_Openmode` [openmode](#)
 - typedef `traits_type::pos_type` [pos_type](#)
 - typedef `int` **`seek_dir`**
 - typedef `_Ios_Seekdir` [seekdir](#)
 - typedef `std::streamoff` **`streamoff`**
 - typedef `std::streampos` **`streampos`**
 - typedef `_Traits` [traits_type](#)
-
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __-`
`num_get_type`

Public Member Functions

- [basic_ostringstream](#) (`ios_base::openmode __mode=ios_base::out`)
- [basic_ostringstream](#) (`const __string_type &__str, ios_base::openmode __-`
`mode=ios_base::out`)
- [~basic_ostringstream](#) ()
- `const locale & _M_getloc` () const
- `void _M_setstate` (`iostate __state`)
- `bool bad` () const
- `void clear` (`iostate __state=goodbit`)
- [basic_ios & copyfmt](#) (`const basic_ios &__rhs`)
- `bool eof` () const
- [iostate exceptions](#) () const
- `void exceptions` (`iostate __except`)
- `bool fail` () const
- [char_type fill](#) () const
- [char_type fill](#) (`char_type __ch`)
- [fmtflags flags](#) (`fmtflags __fmtfl`)
- [fmtflags flags](#) () const
- [__ostream_type & flush](#) ()
- [locale getloc](#) () const
- `bool good` () const
- [locale imbue](#) (`const locale &__loc`)
- `long & iword` (`int __ix`)
- `char narrow` (`char_type __c, char __dfault`) const
- [streamsize precision](#) () const
- [streamsize precision](#) (`streamsize __prec`)
- `void *& pword` (`int __ix`)

- `__stringbuf_type * rdbuf () const`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
- `ios_base::rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `void setstate (ios_base::iostate __state)`
- `void str (const __string_type & __s)`
- `__string_type str () const`
- `pos_type tellp ()`
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
- `basic_ostream< _CharT, _Traits > * tie () const`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`

- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`

- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`

Static Public Member Functions

- static `bool sync_with_stdio (bool __sync=true)`
- static `int xalloc () throw ()`

Static Public Attributes

- static `const fmtflags adjustfield`
- static `const openmode app`
- static `const openmode ate`
- static `const iostate badbit`
- static `const fmtflags basefield`
- static `const seekdir beg`
- static `const openmode binary`
- static `const fmtflags boolalpha`
- static `const seekdir cur`
- static `const fmtflags dec`

- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- template<typename _ValueT >
[__ostream_type](#) & [_M_insert](#) (_ValueT __v)
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

Friends

- `class sentry`

5.373.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_ostringstream<_CharT, _Traits, _Alloc>
```

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 366 of file `sstream`.

5.373.2 Member Typedef Documentation

5.373.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ostream<_CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 71 of file `ostream`.

5.373.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits>::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, and `std::basic_istream<char>`.

Definition at line 86 of file `basic_ios.h`.

5.373.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 70 of file `ostream`.

5.373.2.4 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _CharT std::basic_ostringstream<_CharT, _Traits, _Alloc
>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 370 of file `sstream`.

5.373.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`
[inherited]

The type of an event callback function.

Parameters

- event* One of the members of the event enum.
- ios_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.373.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1957

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.373.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>`
`typedef traits_type::int_type std::basic_ostringstream<_CharT,`
`_Traits, _Alloc>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Definition at line 375 of file `sstream`.

5.373.2.8 `typedef _Ios_Iostate std::ios_base::iostate` **[inherited]**

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.373.2.9 `template<typename _CharT, typename _Traits, typename _Alloc>`
`typedef traits_type::off_type std::basic_ostringstream<_CharT,`
`_Traits, _Alloc>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Definition at line 377 of file sstream.

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 369 of file ios_base.h.

```
5.373.2.11 template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::pos_type std::basic_ostringstream< _CharT,
_Traits, _Alloc >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Definition at line 376 of file sstream.

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

5.373 `std::basic_ostringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1959

Definition at line 401 of file `ios_base.h`.

5.373.2.13 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _Traits std::basic_ostringstream< _CharT, _Traits, _Alloc
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Definition at line 371 of file `sstream`.

5.373.3 Member Enumeration Documentation

5.373.3.1 `enum std::ios_base::event` **[inherited]**

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

5.373.4 Constructor & Destructor Documentation

5.373.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc
>::basic_ostringstream (ios_base::openmode __mode =
ios_base::out) [inline, explicit]`

Default constructor starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 402 of file `sstream`.

5.373.4.2 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc>::basic_ostringstream (const __string_type & __str, ios_base::openmode __mode = ios_base::out) [inline, explicit]`

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes *sb* using *str* and `mode|out`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 420 of file `sstream`.

5.373.4.3 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc>::~~basic_ostringstream () [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 431 of file `sstream`.

5.373.5 Member Function Documentation

5.373.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get<`

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1961

_CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

5.373.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream< _CharT, _Traits >::_M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.373.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.373.5.4 `template<typename _CharT, typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.373.5.5 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.373.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

5.373 `std::basic_ostringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1963

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.373.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions () const` [inline, inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.373.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate __except)` [inline, inherited]

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");
```

```

std::cerr << "Setting badbit\n";
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file `basic_ios.h`.

5.373.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const` **[inline, inherited]**

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.373.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const` **[inline, inherited]**

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1965

5.373.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.373.5.12 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.373.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.373.5.14 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.373.5.15 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.373.5.16 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good () const [inline, inherited]`

Fast error checking.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1967

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 170 of file basic_ios.h.

5.373.5.17 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.373.5.18 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.373.5.19 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.373.5.20 `template<typename _CharT, typename _Traits> char
std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.373.5.21 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits >::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

5.373 std::basic_ostringstream<_CharT, _Traits, _Alloc> Class Template Reference 1969

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.373.5.22 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.373.5.23 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (long __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file `ostream`.

5.373.5.24 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.373.5.25 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (
__ostream_type &(*)(__ostream_type &) __pf) [inline,
inherited]`

Interface for manipulators.

Manipulators such as [std::endl](#) and [std::hex](#) use these functions in constructs like "std::cout << std::endl". For more information, see the [iomanip](#) header.

Definition at line 108 of file ostream.

5.373.5.26 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.373.5.27 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1971

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.373.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * __p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.373.5.29 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as [std::endl](#) and [std::hex](#) use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file ostream.

5.373.5.30 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ostream<_CharT, _Traits>::rdbuf()`, and `std::basic_ostream<_CharT, _Traits>::setstate()`.

5.373.5.31 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.373.5.32 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1973

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file `ostream`.

5.373.5.33 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n) [inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.373.5.34 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

5.373.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(unsigned int __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 191 of file ostream.

```
5.373.5.36  template<typename _CharT, typename _Traits> __ostream_type&
              std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned
              long __n ) [inline, inherited]
```

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 169 of file ostream.

```
5.373.5.37  template<typename _CharT , typename _Traits > basic_ostream<
              _CharT, _Traits > & std::basic_ostream< _CharT, _Traits
              >::operator<< ( short __n ) [inherited]
```

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

5.373 std::basic_ostream< _CharT, _Traits, _Alloc > Class Template Reference 1975

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

5.373.5.38 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.373.5.39 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.373.5.40 `streamsize std::ios_base::precision (streamsize __prec)
[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file `ios_base.h`.

5.373.5.41 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

5.373.5.42 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

**this*

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1977

5.373.5.43 void*& std::ios_base::pword (int __ix) [inline, inherited]

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.373.5.44 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.373.5.45 `template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_ostringstream<_CharT, _Traits,
_Alloc>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 442 of file `sstream`.

5.373.5.46 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate () const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.373.5.47 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1979

5.373.5.48 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.373.5.49 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.373.5.50 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask)`
[inline, inherited]

Setting new format flags.

Parameters

fmtfl Additional flags to set.
mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.373.5.51 `fmtflags std::ios_base::setf (fmtflags __fmtfl)` **[inline, inherited]**

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.373.5.52 `template<typename _CharT, typename _Traits> void`
`std::basic_ios< _CharT, _Traits >::setstate (iostate __state)`
[inline, inherited]

Sets additional flags in the error state.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1981

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.373.5.53 `template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc
>::str () const [inline]`

Copying out the string buffer.

Returns

`rddbuf ()->str ()`

Definition at line 450 of file `sstream`.

Referenced by `std::operator<<()`.

5.373.5.54 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (
const __string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Calls `rddbuf ()->str (s)`.

Definition at line 460 of file `sstream`.

5.373.5.55 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.373.5.56 `template<typename _CharT, typename _Traits> basic_ostream<
_CharT, _Traits>::pos_type std::basic_ostream< _CharT, _Traits
>::tellp () [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits>::rddbuf()`.

5.373.5.57 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits
>::tie () const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1983

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

```
5.373.5.58  template<typename _CharT, typename _Traits>
              basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits
              >::tie ( basic_ostream< _CharT, _Traits > * __tiestr ) [inline,
              inherited]
```

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

```
5.373.5.59  void std::ios_base::unsetf ( fmtflags __mask ) [inline,
              inherited]
```

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

```
5.373.5.60  template<typename _CharT, typename _Traits> char_type
              std::basic_ios<_CharT, _Traits >::widen ( char __c ) const
              [inline, inherited]
```

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.373.5.61 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.373.5.62 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template
Reference **1985**

5.373.5.63 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::write (const char_type *
__s, streamsize __n) [inherited]`

Character string insertion.

Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.373.5.64 `static int std::ios_base::xalloc () throw () [static,
inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.373.6 Member Data Documentation

5.373.6.1 `const fmtflags std::ios_base::adjustfield` `[static, inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.373.6.2 `const openmode std::ios_base::app` `[static, inherited]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.373.6.3 `const openmode std::ios_base::ate` `[static, inherited]`

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

5.373.6.4 `const iostate std::ios_base::badbit` `[static, inherited]`

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1987

5.373.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.373.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

5.373.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.373.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.373.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.373.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.373.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.373.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsomewhat(), and std::ws().

5.373.6.13 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.373.6.14 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1989

5.373.6.15 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file `ios_base.h`.

5.373.6.16 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.373.6.17 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.373.6.18 const openmode std::ios_base::in [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.373.6.19 `const fmtflags std::ios_base::internal` `[static, inherited]`

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

5.373.6.20 `const fmtflags std::ios_base::left` `[static, inherited]`

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.373.6.21 `const fmtflags std::ios_base::oct` `[static, inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.373.6.22 `const openmode std::ios_base::out` `[static, inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.373.6.23 `const fmtflags std::ios_base::right` `[static, inherited]`

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file `ios_base.h`.

5.373.6.24 `const fmtflags std::ios_base::scientific` `[static, inherited]`

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

5.373 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1991

5.373.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.373.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios_base.h.

5.373.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios_base.h.

5.373.6.28 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 308 of file ios_base.h.

5.373.6.29 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 389 of file ios_base.h.

5.373.6.30 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file ios_base.h.

5.373.6.31 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following file:

- [sstream](#)

5.374 `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference

Public Types

- typedef [regex_constants::syntax_option_type](#) **flag_type**
- typedef [_Rx_traits::locale_type](#) **locale_type**
- typedef [_Rx_traits::string_type](#) **string_type**
- typedef [_Ch_type](#) **value_type**

Public Member Functions

- [basic_regex](#) ()
- [basic_regex](#) (const [_Ch_type](#) *__p, [flag_type](#) __f=[regex_constants::ECMAScript](#))
- [basic_regex](#) (const [basic_regex](#) &__rhs)
- [basic_regex](#) ([initializer_list](#)< [_Ch_type](#) > __l, [flag_type](#) __f=[regex_constants::ECMAScript](#))
- [basic_regex](#) (const [basic_regex](#) &&__rhs)
- [basic_regex](#) (const [_Ch_type](#) *__p, [std::size_t](#) __len, [flag_type](#) __f)
- [template](#)<[typename](#) [_Ch_traits](#), [typename](#) [_Ch_alloc](#) > [basic_regex](#) (const [std::basic_string](#)< [_Ch_type](#), [_Ch_traits](#), [_Ch_alloc](#) > &__s, [flag_type](#) __f=[regex_constants::ECMAScript](#))
- [template](#)<[typename](#) [_InputIterator](#) > [basic_regex](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [flag_type](#) __f=[regex_constants::ECMAScript](#))
- [~basic_regex](#) ()
- const [__regex::__AutomatonPtr](#) & [_M_get_automaton](#) () const
- [template](#)<[typename](#) [_InputIterator](#) > [basic_regex](#) & [assign](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [flag_type](#) __flags=[regex_constants::ECMAScript](#))
- [basic_regex](#) & [assign](#) (const [_Ch_type](#) *__p, [flag_type](#) __flags=[regex_constants::ECMAScript](#))
- [basic_regex](#) & [assign](#) (const [_Ch_type](#) *__p, [std::size_t](#) __len, [flag_type](#) __flags)
- [template](#)<[typename](#) [_Ch_traits](#), [typename](#) [_Allocator](#) > [basic_regex](#) & [assign](#) (const [basic_string](#)< [_Ch_type](#), [_Ch_traits](#), [_Allocator](#) > &__s, [flag_type](#) __f=[regex_constants::ECMAScript](#))

- [basic_regex](#) & [assign](#) (initializer_list<_Ch_type> __l, flag_type __f=[regex_constants::ECMAScript](#))
- [basic_regex](#) & [assign](#) (const [basic_regex](#) &__rhs)
- [basic_regex](#) & [assign](#) ([basic_regex](#) &&__rhs)
- flag_type [flags](#) () const
- locale_type [getloc](#) () const
- locale_type [imbue](#) (locale_type __loc)
- unsigned int [mark_count](#) () const
- [basic_regex](#) & [operator=](#) (const _Ch_type *__p)
- [basic_regex](#) & [operator=](#) (const [basic_regex](#) &__rhs)
- [basic_regex](#) & [operator=](#) ([basic_regex](#) &&__rhs)
- template<typename _Ch_type, traits, typename _Allocator>
[basic_regex](#) & [operator=](#) (const [basic_string](#)<_Ch_type, _Ch_type_traits, _Allocator> &__s)
- void [swap](#) ([basic_regex](#) &__rhs)

Static Public Attributes

Constants

std [28.8.1](1)

Todo

These should be constexpr.

- static const [regex_constants::syntax_option_type](#) **icase**
- static const [regex_constants::syntax_option_type](#) **nosubs**
- static const [regex_constants::syntax_option_type](#) **optimize**
- static const [regex_constants::syntax_option_type](#) **collate**
- static const [regex_constants::syntax_option_type](#) **ECMAScript**
- static const [regex_constants::syntax_option_type](#) **basic**
- static const [regex_constants::syntax_option_type](#) **extended**
- static const [regex_constants::syntax_option_type](#) **awk**
- static const [regex_constants::syntax_option_type](#) **grep**
- static const [regex_constants::syntax_option_type](#) **egrep**

Protected Attributes

- [__regex::AutomatonPtr](#) **_M_automaton**
- flag_type **_M_flags**
- [_Rx_traits](#) **_M_traits**

5.374.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::basic_regex< _Ch_type, _Rx_traits >
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 340 of file `regex.h`.

5.374.2 Constructor & Destructor Documentation

```
5.374.2.1 template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex( ) [inline]
```

Constructs a basic regular expression that does not match any character sequence.

Definition at line 382 of file `regex.h`.

```
5.374.2.2 template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex ( const _Ch_type * __p, flag_type __f =
regex_constants::ECMAScript ) [inline, explicit]
```

Constructs a basic regular expression from the sequence `[p, p + char_traits<_Ch_type>::length(p))` interpreted according to the flags in `f`.

Parameters

p A pointer to the start of a C-style null-terminated string containing a regular expression.

f Flags indicating the syntax rules and options.

Exceptions

[*regex_error*](#) if `p` is not a valid regular expression.

Definition at line 400 of file `regex.h`.

5.374 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference 1995

5.374.2.3 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (const _Ch_type * __p, std::size_t __len, flag_type
__f) [inline]`

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in f.

Parameters

- p* A pointer to the start of a string containing a regular expression.
- len* The length of the string containing the regular expression.
- f* Flags indicating the syntax rules and options.

Exceptions

- regex_error* if p is not a valid regular expression.

Definition at line 418 of file regex.h.

5.374.2.4 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (const basic_regex< _Ch_type, _Rx_traits > &
__rhs) [inline]`

Copy-constructs a basic regular expression.

Parameters

- rhs* A regex object.

Definition at line 428 of file regex.h.

5.374.2.5 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (const basic_regex< _Ch_type, _Rx_traits > &&
__rhs) [inline]`

Move-constructs a basic regular expression.

Parameters

- rhs* A regex object.

Definition at line 438 of file regex.h.

```

5.374.2.6 template<typename _Ch_type, typename _Rx_traits =
        regex_traits<_Ch_type>> template<typename _Ch_traits ,
        typename _Ch_alloc > std::basic_regex< _Ch_type, _Rx_traits
        >::basic_regex ( const std::basic_string< _Ch_type, _Ch_traits,
        _Ch_alloc > & __s, flag_type __f= regex_constants::ECMAScript )
        [inline, explicit]

```

Constructs a basic regular expression from the string *s* interpreted according to the flags in *f*.

Parameters

s A string containing a regular expression.

f Flags indicating the syntax rules and options.

Exceptions

regex_error if *s* is not a valid regular expression.

Definition at line 454 of file regex.h.

```

5.374.2.7 template<typename _Ch_type, typename _Rx_traits =
        regex_traits<_Ch_type>> template<typename _InputIterator
        > std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
        _InputIterator __first, _InputIterator __last, flag_type __f=
        regex_constants::ECMAScript ) [inline]

```

Constructs a basic regular expression from the range [*first*, *last*) interpreted according to the flags in *f*.

Parameters

first The start of a range containing a valid regular expression.

last The end of a range containing a valid regular expression.

f The format flags of the regular expression.

Exceptions

regex_error if [*first*, *last*) is not a valid regular expression.

Definition at line 476 of file regex.h.

5.374 std::basic_regex<_Ch_type, _Rx_traits> Class Template Reference 1997

5.374.2.8 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits
>::basic_regex (initializer_list<_Ch_type> __l, flag_type __f =
regex_constants::ECMAScript) [inline]`

Constructs a basic regular expression from an initializer list.

Parameters

l The initializer list.

f The format flags of the regular expression.

Exceptions

[*regex_error*](#) if *l* is not a valid regular expression.

Definition at line 490 of file regex.h.

5.374.2.9 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits
>::~~basic_regex () [inline]`

Destroys a basic regular expression.

Definition at line 500 of file regex.h.

5.374.3 Member Function Documentation

5.374.3.1 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits>::assign (const basic_regex<_Ch_type,
_Rx_traits> & __rhs) [inline]`

the real assignment operator.

Parameters

rhs Another regular expression object.

Definition at line 546 of file regex.h.

References std::basic_regex<_Ch_type, _Rx_traits>::swap().

Referenced by std::basic_regex<_Ch_type, _Rx_traits>::operator=().

5.374.3.2 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (const _Ch_type * __p, std::size_t
 __len, flag_type __flags) [inline]`

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

Parameters

- p* A pointer to a C-style string containing a regular expression pattern.
- len* The length of the regular expression pattern string.
- flags* Syntax option flags.

Exceptions

- [*regex_error*](#) if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If [*regex_error*](#) is thrown, **this* remains unchanged.

Definition at line 598 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

5.374.3.3 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> template<typename _Ch_type, traits_type,
 typename _Allocator > basic_regex& std::basic_regex< _Ch_type,
 _Rx_traits >::assign (const basic_string< _Ch_type, _Ch_type_traits,
 _Allocator > & __s, flag_type __f = regex_constants::ECMAScript
) [inline]`

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

Parameters

- s* A string containing a regular expression pattern.
- flags* Syntax option flags.

Exceptions

- [*regex_error*](#) if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If [*regex_error*](#) is thrown, **this* remains unchanged.

Definition at line 614 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

5.374 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference 1999

5.374.3.4 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> template<typename _InputIterator >
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
_InputIterator __first, _InputIterator __last, flag_type __flags =
regex_constants::ECMAScript) [inline]`

Assigns a new regular expression to a regex object.

Parameters

first The start of a range containing a valid regular expression.

last The end of a range containing a valid regular expression.

flags Syntax option flags.

Exceptions

regex_error if p does not contain a valid regular expression pattern interpreted according to flags. If **regex_error** is thrown, the object remains unchanged.

Definition at line 637 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

5.374.3.5 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::assign (initializer_list< _Ch_type > __l,
flag_type __f= regex_constants::ECMAScript) [inline]`

Assigns a new regular expression to a regex object.

Parameters

l An initializer list representing a regular expression.

flags Syntax option flags.

Exceptions

regex_error if l does not contain a valid regular expression pattern interpreted according to flags. If **regex_error** is thrown, the object remains unchanged.

Definition at line 652 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

5.374.3.6 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (basic_regex< _Ch_type, _Rx_traits
 > && __rhs) [inline]`

The move-assignment operator.

Parameters

rhs Another regular expression object.

Definition at line 559 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

5.374.3.7 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (const _Ch_type * __p, flag_type
 __flags = regex_constants::ECMAScript) [inline]`

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

Parameters

p A pointer to a C-style null-terminated string containing a regular expression pattern.

flags Syntax option flags.

Exceptions

[*regex_error*](#) if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If [*regex_error*](#) is thrown, *this remains unchanged.

Definition at line 580 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

5.374.3.8 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> flag_type std::basic_regex< _Ch_type,
 _Rx_traits >::flags () const [inline]`

Gets the flags used to construct the regular expression or in the last call to [`assign\(\)`](#).

Definition at line 670 of file regex.h.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::operator=()`.

5.374 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference 2001

5.374.3.9 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type,
_Rx_traits >::getloc () const [inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 688 of file regex.h.

5.374.3.10 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type,
_Rx_traits >::imbue (locale_type __loc) [inline]`

Imbues the regular expression object with the given locale.

Parameters

loc A locale.

Definition at line 680 of file regex.h.

5.374.3.11 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> unsigned int std::basic_regex<
_Ch_type, _Rx_traits >::mark_count () const [inline]`

Gets the number of marked subexpressions within the regular expression.

Definition at line 662 of file regex.h.

5.374.3.12 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::operator= (basic_regex< _Ch_type,
_Rx_traits > && __rhs) [inline]`

Move-assigns one regular expression to another.

Definition at line 514 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

5.374.3.13 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::operator= (const basic_regex< _Ch_type,
_Rx_traits > & __rhs) [inline]`

Assigns one regular expression to another.

Definition at line 507 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

5.374.3.14 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> template<typename _Ch_type, _Rx_traits,
typename _Allocator > basic_regex& std::basic_regex< _Ch_type,
_Rx_traits >::operator= (const basic_string< _Ch_type,
_Ch_type, _Allocator > & __s) [inline]`

Replaces a regular expression with a new one constructed from a string.

Parameters

A pointer to a string containing a regular expression.

Definition at line 536 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

5.374.3.15 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::operator= (const _Ch_type * __p)
[inline]`

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

A pointer to the start of a null-terminated C-style string containing a regular expression.

Definition at line 525 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

5.374.3.16 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> void std::basic_regex< _Ch_type,
_Rx_traits >::swap (basic_regex< _Ch_type, _Rx_traits > &
__rhs) [inline]`

Swaps the contents of two regular expression objects.

Parameters

rhs Another regular expression object.

Definition at line 698 of file regex.h.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`, and `std::swap()`.

The documentation for this class was generated from the following file:

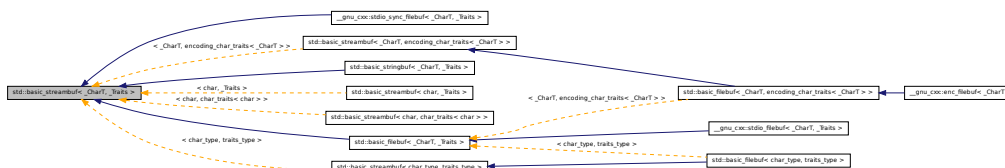
- [regex.h](#)

5.375 std::basic_streambuf<_CharT, _Traits> Class Template Reference

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Inheritance diagram for `std::basic_streambuf<_CharT, _Traits>`:



Public Types

- typedef _CharT char_type
 - typedef _Traits traits_type
 - typedef traits_type::int_type int_type
 - typedef traits_type::pos_type pos_type
 - typedef traits_type::off_type off_type
-
- typedef basic_streambuf< char_type, traits_type > __streambuf_type

Public Member Functions

- `streamsize in_avail ()`
- `int_type sbumpc ()`

- [int_type sgetc \(\)](#)
- [streamsize sgetn \(char_type *__s, streamsize __n\)](#)
- [int_type snextc \(\)](#)
- [int_type sputbackc \(char_type __c\)](#)
- [int_type sputc \(char_type __c\)](#)
- [streamsize sputn \(const char_type *__s, streamsize __n\)](#)
- [void stoss \(\)](#)
- [int_type sungetc \(\)](#)

Protected Member Functions

- [basic_streambuf \(\)](#)
 - [void gbump \(int __n\)](#)
 - [virtual void imbue \(const locale &\)](#)
 - [virtual int_type overflow \(int_type=traits_type::eof\(\)\)](#)
 - [virtual int_type pbackfail \(int_type=traits_type::eof\(\)\)](#)
 - [void pbump \(int __n\)](#)
 - [virtual pos_type seekoff \(off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out\)](#)
 - [virtual pos_type seekpos \(pos_type, ios_base::openmode=ios_base::in|ios_base::out\)](#)
 - [virtual basic_streambuf< char_type, _Traits > * setbuf \(char_type *, streamsize\)](#)
 - [void setg \(char_type *__gbeg, char_type *__gnext, char_type *__gend\)](#)
 - [void setp \(char_type *__pbeg, char_type *__pend\)](#)
 - [virtual streamsize showmanyc \(\)](#)
 - [virtual int sync \(\)](#)
 - [virtual int_type uflow \(\)](#)
 - [virtual int_type underflow \(\)](#)
 - [virtual streamsize xsgetn \(char_type *__s, streamsize __n\)](#)
 - [virtual streamsize xsputn \(const char_type *__s, streamsize __n\)](#)
-
- [char_type * eback \(\) const](#)
 - [char_type * gptr \(\) const](#)
 - [char_type * egptr \(\) const](#)
-
- [char_type * pbase \(\) const](#)
 - [char_type * pptr \(\) const](#)
 - [char_type * epptr \(\) const](#)

Friends

- template<bool _IsMove, typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)
 - streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *, bool &)
 - class basic_ios< char_type, traits_type >
 - class basic_istream< char_type, traits_type >
 - class basic_ostream< char_type, traits_type >
 - template<typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)
 - template<typename _CharT2, typename _Traits2, typename _Alloc >
basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)
 - class istreambuf_iterator< char_type, traits_type >
 - template<typename _CharT2, typename _Traits2, typename _Alloc >
basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)
 - template<typename _CharT2, typename _Traits2 >
basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, _CharT2 *)
 - class ostreambuf_iterator< char_type, traits_type >
-
- char_type * _M_in_beg
 - char_type * _M_in_cur
 - char_type * _M_in_end
 - char_type * _M_out_beg
 - char_type * _M_out_cur
 - char_type * _M_out_end
 - locale _M_buf_locale
 - virtual ~basic_streambuf ()
 - locale pubimbue (const locale &__loc)
 - locale getloc () const
 - __streambuf_type * pubsetbuf (char_type *__s, streamsize __n)
 - pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
 - pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
 - int pubsync ()

5.375.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_streambuf< _  
_CharT, _Traits >
```

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output. Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:
 - The controlled input sequence can be not readable.
 - The controlled output sequence can be not writable.
 - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
 - The controlled sequences can support operations *directly* to or from associated sequences.
 - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
 - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
 - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
 - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
 - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.

- If *xnext* is not a null pointer and *xnext* < *xend* for an output sequence, then a *write position* is available. In this case, **xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
- If *xnext* is not a null pointer and *xbeg* < *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext*[-1] shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
- If *xnext* is not a null pointer and *xnext* < *xend* for an input sequence, then a *read position* is available. In this case, **xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 115 of file `streambuf`.

5.375.2 Member Typedef Documentation

5.375.2.1 `template<typename _CharT, typename _Traits> typedef
basic_streambuf<char_type, traits_type> std::basic_streambuf<
_CharT, _Traits >::__streambuf_type`

This is a non-standard type.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 133 of file `streambuf`.

5.375.2.2 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_streambuf< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 124 of file `streambuf`.

5.375.2.3 `template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 126 of file `streambuf`.

5.375.2.4 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 128 of file `streambuf`.

5.375.2.5 `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::enc_filebuf< _CharT >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 127 of file `streambuf`.

**5.375.2.6 template<typename _CharT, typename _Traits> typedef _Traits
std::basic_streambuf< _CharT, _Traits >::traits_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#), [__gnu_cxx::enc_filebuf< _CharT >](#), [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 125 of file streambuf.

5.375.3 Constructor & Destructor Documentation

**5.375.3.1 template<typename _CharT, typename _Traits> virtual
std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf ()
[inline, virtual]**

Destructor deallocates no buffer space.

Definition at line 193 of file streambuf.

**5.375.3.2 template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf ()
[inline, protected]**

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the [basic_streambuf](#) class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 441 of file streambuf.

5.375.4 Member Function Documentation

5.375.4.1 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eback () const
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.375.4.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::egptr () const
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.375.4.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eptr () const
[inline, protected]`

Access to the put area.

5.375 std::basic_streambuf< _CharT, _Traits > Class Template Reference 2011

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.375.4.4 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.375.4.5 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::getloc () const
[inline]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.375.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.375.4.7 `template<typename _CharT, typename _Traits> virtual void
std::basic_streambuf< _CharT, _Traits >::imbue (const locale &)
[inline, protected, virtual]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 554 of file streambuf.

5.375.4.8 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::in_avail () [inline]`

Looking ahead into the stream.

Returns

The number of characters available.

5.375 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2013

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file `streambuf`.

5.375.4.9 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::overflow (int_type =
traits_type::eof()) [inline, protected, virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output `streambuf` can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 746 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.375.4.10 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 702 of file streambuf.

5.375.4.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pbase () const
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`.

5.375.4.12 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::pbump (int __n)
[inline, protected]`

Moving the write position.

Parameters

n The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xspn()`.

5.375.4.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.375.4.14 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

5.375.4.15 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

5.375.4.16 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.375.4.17 `template<typename _CharT, typename _Traits>
__streambuf_type* std::basic_streambuf< _CharT, _Traits
>::pubsetbuf (char_type * __s, streamsize __n) [inline]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.375.4.18 `template<typename _CharT, typename _Traits> int
std::basic_streambuf< _CharT, _Traits >::pubsync ()
[inline]`

Entry point for `imbue()`.

Parameters

loc The new locale.

5.375 std::basic_streambuf<_CharT, _Traits> Class Template Reference 2017

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 249 of file streambuf.

Referenced by std::basic_istream<_CharT, _Traits>::sync().

5.375.4.19 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sbumpc() [inline]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file streambuf.

Referenced by std::basic_istream<_CharT, _Traits>::getline(), std::basic_istream<_CharT, _Traits>::ignore(), and std::istreambuf_iterator<_CharT, _Traits>::operator++().

5.375.4.20 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file streambuf.

5.375.4.21 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.375.4.22 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT, _Traits >::setbuf (char_type *, streamsize) [inline,
protected, virtual]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

5.375.4.23 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setg (char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected]`

Setting the three read area pointers.

Parameters

gbeg A pointer.

gnext A pointer.

gend A pointer.

5.375 std::basic_streambuf<_CharT, _Traits> Class Template Reference 2019

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

5.375.4.24 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setp (char_type *
__pbeg, char_type * __pend) [inline, protected]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.375.4.25 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sgetc () [inline]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.375.4.26 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sgetn (char_type * __s,
streamsize __n) [inline]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.375.4.27 `template<typename _CharT, typename _Traits> virtual streamsize
std::basic_streambuf< _CharT, _Traits >::showmanyc ()
[inline, protected, virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 627 of file `streambuf`.

5.375.4.28 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::snextc () [inline]`

Getting the next character.

Returns

The next character, or eof.

5.375 std::basic_streambuf<_CharT, _Traits> Class Template Reference 2021

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.375.4.29 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc (char_type
__c) [inline]`

Pushing characters back into the input stream.

Parameters

`c` The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but `c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `c`.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

5.375.4.30 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc (char_type __c)
[inline]`

Entry point for all single-character output functions.

Parameters

`c` A character to output.

Returns

`c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `c` in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

5.375.4.31 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sputn (const char_type
* __s, streamsize __n) [inline]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file streambuf.

5.375.4.32 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::stossc () [inline]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file streambuf.

5.375.4.33 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

5.375 std::basic_streambuf< _CharT, _Traits > Class Template Reference 2023

Definition at line 375 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

5.375.4.34 `template<typename _CharT, typename _Traits> virtual int
std::basic_streambuf< _CharT, _Traits >::sync (void)
[inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 605 of file streambuf.

5.375.4.35 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 678 of file streambuf.

5.375.4.36 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::underflow ()
[inline, protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 665 of file streambuf.

5.375.4.37 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type *
__s, streamsize __n) [protected, virtual]`

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

5.375 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2025

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

5.375.4.38 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::xsputn (const char_type
* __s, streamsize __n) [protected, virtual]`

Multiple character insertion.

Parameters

s A buffer area.

n Maximum number of characters to write.

Returns

The number of characters written.

Writes `s[0]` through `s[n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.375.5 Member Data Documentation

5.375.5.1 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale
[protected]`

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

5.375.5.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_beg
[protected]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file streambuf.

5.375.5.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_cur
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file streambuf.

5.375.5.4 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_end
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.375.5.5 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.375.5.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.375.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

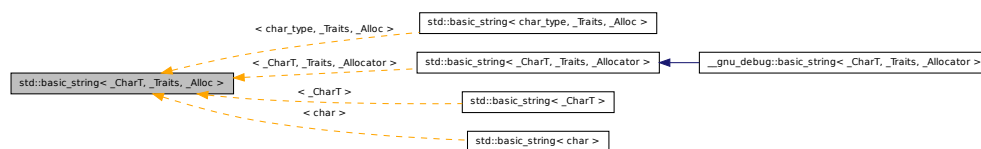
The documentation for this class was generated from the following files:

- [streambuf](#)
- [streambuf.tcc](#)

5.376 `std::basic_string< _CharT, _Traits, _Alloc >` Class Template Reference

Managing sequences of characters and character-like objects.

Inheritance diagram for `std::basic_string< _CharT, _Traits, _Alloc >`:

**Public Types**

- typedef `_Alloc allocator_type`
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic_string > const_iterator`
- typedef `_CharT_alloc_type::const_pointer const_pointer`
- typedef `_CharT_alloc_type::const_reference const_reference`
- typedef `std::reverse_iterator< const_iterator > const_reverse_iterator`
- typedef `_CharT_alloc_type::difference_type difference_type`
- typedef `__gnu_cxx::__normal_iterator< pointer, basic_string > iterator`
- typedef `_CharT_alloc_type::pointer pointer`
- typedef `_CharT_alloc_type::reference reference`

- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_CharT_alloc_type::size_type` **size_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits::char_type` **value_type**

Public Member Functions

- `basic_string()`
- `basic_string(const _Alloc &__a)`
- `basic_string(const basic_string &__str, size_type __pos, size_type __n=npos)`
- `basic_string(size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `basic_string(basic_string &&__str)`
- `basic_string(const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `basic_string(initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())`
- template<class _InputIterator >
`basic_string(_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())`
- `basic_string(const basic_string &__str)`
- `basic_string(const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())`
- `basic_string(const _CharT *__s, const _Alloc &__a=_Alloc())`
- `~basic_string()`
- template<typename _InIterator >
`_CharT * _S_construct(_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)`
- `basic_string & append(const basic_string &__str)`
- `basic_string & append(const basic_string &__str, size_type __pos, size_type __n)`
- `basic_string & append(const _CharT *__s, size_type __n)`
- `basic_string & append(const _CharT *__s)`
- `basic_string & append(size_type __n, _CharT __c)`
- `basic_string & append(initializer_list< _CharT > __l)`
- template<class _InputIterator >
`basic_string & append(_InputIterator __first, _InputIterator __last)`
- `basic_string & assign(const _CharT *__s)`
- `basic_string & assign(size_type __n, _CharT __c)`
- template<class _InputIterator >
`basic_string & assign(_InputIterator __first, _InputIterator __last)`
- `basic_string & assign(initializer_list< _CharT > __l)`
- `basic_string & assign(const basic_string &__str)`
- `basic_string & assign(basic_string &&__str)`
- `basic_string & assign(const basic_string &__str, size_type __pos, size_type __n)`

- [basic_string](#) & [assign](#) (const _CharT *__s, size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [at](#) (size_type __n)
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const _CharT * [c_str](#) () const
- size_type [capacity](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- int [compare](#) (size_type __pos, size_type __n1, const _CharT *__s) const
- int [compare](#) (const _CharT *__s) const
- int [compare](#) (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int [compare](#) (size_type __pos, size_type __n, const [basic_string](#) &__str) const
- int [compare](#) (size_type __pos1, size_type __n1, const [basic_string](#) &__str, size_type __pos2, size_type __n2) const
- int [compare](#) (const [basic_string](#) &__str) const
- size_type [copy](#) (_CharT *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- const _CharT * [data](#) () const
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- [basic_string](#) & [erase](#) (size_type __pos=0, size_type __n=[npos](#))
- iterator [erase](#) (iterator __position)
- iterator [erase](#) (iterator __first, iterator __last)
- size_type [find](#) (_CharT __c, size_type __pos=0) const
- size_type [find](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find](#) (const [basic_string](#) &__str, size_type __pos=0) const
- size_type [find](#) (const _CharT *__s, size_type __pos=0) const
- size_type [find_first_not_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_first_not_of](#) (_CharT __c, size_type __pos=0) const
- size_type [find_first_not_of](#) (const [basic_string](#) &__str, size_type __pos=0) const
- size_type [find_first_not_of](#) (const _CharT *__s, size_type __pos=0) const
- size_type [find_first_of](#) (_CharT __c, size_type __pos=0) const
- size_type [find_first_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_first_of](#) (const _CharT *__s, size_type __pos=0) const

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2031

- size_type [find_first_of](#) (const [basic_string](#) &__str, size_type __pos=0) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_last_not_of](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [find_last_not_of](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const
- void [insert](#) (iterator __p, size_type __n, _CharT __c)
- template<class _InputIterator>
void [insert](#) (iterator __p, _InputIterator __beg, _InputIterator __end)
- void [insert](#) (iterator __p, [initializer_list](#)<_CharT> __l)
- [basic_string](#) & [insert](#) (size_type __pos1, const [basic_string](#) &__str, size_type __pos2, size_type __n)
- [basic_string](#) & [insert](#) (size_type __pos, const _CharT *__s, size_type __n)
- [basic_string](#) & [insert](#) (size_type __pos, const _CharT *__s)
- iterator [insert](#) (iterator __p, _CharT __c)
- [basic_string](#) & [insert](#) (size_type __pos, size_type __n, _CharT __c)
- [basic_string](#) & [insert](#) (size_type __pos1, const [basic_string](#) &__str)
- size_type [length](#) () const
- size_type [max_size](#) () const
- [basic_string](#) & [operator+=](#) ([initializer_list](#)<_CharT> __l)
- [basic_string](#) & [operator+=](#) (const [basic_string](#) &__str)
- [basic_string](#) & [operator+=](#) (_CharT __c)
- [basic_string](#) & [operator+=](#) (const _CharT *__s)
- [basic_string](#) & [operator=](#) (const _CharT *__s)
- [basic_string](#) & [operator=](#) (_CharT __c)
- [basic_string](#) & [operator=](#) ([initializer_list](#)<_CharT> __l)
- [basic_string](#) & [operator=](#) ([basic_string](#) &&__str)
- [basic_string](#) & [operator=](#) (const [basic_string](#) &__str)
- const_reference [operator\[\]](#) (size_type __pos) const
- reference [operator\[\]](#) (size_type __pos)
- void [push_back](#) (_CharT __c)
- [reverse_iterator](#) [rbegin](#) ()
- const [reverse_iterator](#) [rbegin](#) () const
- [reverse_iterator](#) [rend](#) ()

- [const_reverse_iterator rend](#) () const
- [basic_string & replace](#) (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- [basic_string & replace](#) (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const [basic_string](#) &__str)
- template<class _InputIterator >
[basic_string & replace](#) (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- [basic_string & replace](#) (size_type __pos, size_type __n, const [basic_string](#) &__str)
- [basic_string & replace](#) (size_type __pos, size_type __n1, const _CharT *__s)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- [basic_string & replace](#) (size_type __pos1, size_type __n1, const [basic_string](#) &__str, size_type __pos2, size_type __n2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const _CharT *__s)
- [basic_string & replace](#) (iterator __i1, iterator __i2, [initializer_list](#)< _CharT > __l)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- void [reserve](#) (size_type __res_arg=0)
- void [resize](#) (size_type __n)
- void [resize](#) (size_type __n, _CharT __c)
- size_type [rfind](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [rfind](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [rfind](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [rfind](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- void [shrink_to_fit](#) ()
- size_type [size](#) () const
- [basic_string substr](#) (size_type __pos=0, size_type __n=[npos](#)) const
- void [swap](#) ([basic_string](#) &__s)

Static Public Attributes

- static const size_type [npos](#)

5.376.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_string< _CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Documentation? What's that? Nathan Myers <ncm@cantrip.org>.

A string looks like this:

	<code>[_Rep]</code>
	<code>_M_length</code>
<code>[basic_string<char_type>]</code>	<code>_M_capacity</code>
<code>_M_dataplus</code>	<code>_M_refcount</code>
<code>_M_p -----></code>	unnamed array of <code>char_type</code>

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 105 of file `basic_string.h`.

5.376.2 Constructor & Destructor Documentation

5.376.2.1 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ()
[inline]`

Default constructor creates an empty string.

Definition at line 423 of file `basic_string.h`.

5.376.2.2 `template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const
_Alloc & __a) [explicit]`

Construct an empty string using allocator *a*.

Definition at line 178 of file `basic_string.tcc`.

5.376.2.3 `template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const
basic_string< _CharT, _Traits, _Alloc > & __str)`

Construct string with copy of value of *str*.

Parameters

str Source string.

Definition at line 170 of file `basic_string.tcc`.

5.376.2.4 `template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const
basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos,
size_type __n = npos)`

Construct string as copy of a substring.

Parameters

str Source string.

pos Index of first character to copy from.

n Number of characters to copy (default remainder).

Definition at line 184 of file `basic_string.tcc`.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference

5.376.2.5 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos,
size_type __n, const _Alloc & __a)`

Construct string as copy of a substring.

Parameters

- str* Source string.
- pos* Index of first character to copy from.
- n* Number of characters to copy.
- a* Allocator to use.

Definition at line 194 of file basic_string.tcc.

5.376.2.6 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const
_CharT * __s, size_type __n, const _Alloc & __a = _Alloc())`

Construct string initialized by a character array.

Parameters

- s* Source character array.
- n* Number of characters to copy.
- a* Allocator to use (default is default allocator).

NB: *s* must have at least *n* characters, `'\0'` has no special meaning.

Definition at line 206 of file basic_string.tcc.

5.376.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const
_CharT * __s, const _Alloc & __a = _Alloc())`

Construct string as copy of a C string.

Parameters

- s* Source C string.
- a* Allocator to use (default is default allocator).

Definition at line 213 of file basic_string.tcc.

5.376.2.8 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
size_type __n, _CharT __c, const _Alloc & __a = _Alloc())`

Construct string as multiple characters.

Parameters

- n* Number of characters.
- c* Character to use.
- a* Allocator to use (default is default allocator).

Definition at line 220 of file basic_string.tcc.

5.376.2.9 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
basic_string< _CharT, _Traits, _Alloc > && __str) [inline]`

Move construct string.

Parameters

- str* Source string.

The newly-created string contains the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 493 of file basic_string.h.

5.376.2.10 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
initializer_list< _CharT > __l, const _Alloc & __a = _Alloc())`

Construct string from an initializer list.

Parameters

- l* `std::initializer_list` of characters.
- a* Allocator to use (default is default allocator).

Definition at line 235 of file basic_string.tcc.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2037

5.376.2.11 `template<typename _CharT , typename _Traits , typename
_Alloc> template<typename _InputIterator > std::basic_string<
_CharT, _Traits, _Alloc >::basic_string (_InputIterator __beg,
_InputIterator __end, const _Alloc & __a = _Alloc())`

Construct string as copy of a range.

Parameters

beg Start of range.

end End of range.

a Allocator to use (default is default allocator).

Definition at line 228 of file basic_string.tcc.

5.376.2.12 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ()
[inline]`

Destroy the string instance.

Definition at line 524 of file basic_string.h.

5.376.3 Member Function Documentation

5.376.3.1 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append (const basic_string< _CharT,
_Traits, _Alloc > & __str, size_type __pos, size_type __n)`

Append a substring.

Parameters

str The string to append.

pos Index of the first character of str to append.

n The number of characters to append.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function appends n characters from *str* starting at *pos* to this string. If n is larger than the number of available characters in *str*, the remainder of *str* is appended.

Definition at line 342 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.376.3.2 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<
_CharT, _Traits, _Alloc>::append (const _CharT * __s, size_type
__n)`

Append a C substring.

Parameters

- s The C string to append.
- n The number of characters to append.

Returns

Reference to this string.

Definition at line 298 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.376.3.3 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append
(const _CharT * __s) [inline]`

Append a C string.

Parameters

- s The C string to append.

Returns

Reference to this string.

Definition at line 988 of file `basic_string.h`.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

5.376.3.4 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append (size_type __n, _CharT __c)`

Append multiple characters.

Parameters

n The number of characters to append.

c The character to use.

Returns

Reference to this string.

Appends *n* copies of *c* to this string.

Definition at line 281 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.376.3.5 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append
(initializer_list< _CharT > __l) [inline]`

Append an [initializer_list](#) of characters.

Parameters

l The [initializer_list](#) of characters to append.

Returns

Reference to this string.

Definition at line 1012 of file `basic_string.h`.

Referenced by `std::basic_string< char >::append()`.

5.376.3.6 `template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator > basic_string& std::basic_string<
_CharT, _Traits, _Alloc >::append (_InputIterator __first,
_InputIterator __last) [inline]`

Append a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 1026 of file basic_string.h.

5.376.3.7 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append (const basic_string< _CharT,
_Traits, _Alloc > & __str)`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

Definition at line 325 of file basic_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::collate< _CharT >::do_transform()`, `std::operator+()`, `std::operator>>()`, and `std::basic_string< _CharT, _Traits, _Alloc >::resize()`.

5.376.3.8 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
basic_string< _CharT, _Traits, _Alloc > && __str) [inline]`

Set value to contents of another string.

Parameters

str Source string to use.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference2041

Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1061 of file basic_string.h.

```
5.376.3.9  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
            __pos, size_type __n ) [inline]
```

Set value to a substring of a string.

Parameters

str The string to use.

pos Index of the first character of *str*.

n Number of characters to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1081 of file basic_string.h.

Referenced by std::basic_string< char >::assign().

```
5.376.3.10 template<typename _CharT, typename _Traits , typename _Alloc
            > basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
            _CharT, _Traits, _Alloc >::assign ( const _CharT * __s, size_type
            __n )
```

Set value to a C substring.

Parameters

s The C string to use.

n Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

Definition at line 259 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc >::size().

5.376.3.11 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign
(const _CharT * __s) [inline]`

Set value to contents of a C string.

Parameters

s The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1109 of file basic_string.h.

5.376.3.12 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign
(size_type __n, _CharT __c) [inline]`

Set value to multiple characters.

Parameters

n Length of the resulting string.

c The character to use.

Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1125 of file basic_string.h.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

5.376.3.13 `template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator > basic_string& std::basic_string<
_CharT, _Traits, _Alloc >::assign (_InputIterator __first,
_InputIterator __last) [inline]`

Set value to a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 1138 of file basic_string.h.

5.376.3.14 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign
(initializer_list< _CharT > __l) [inline]`

Set value to an [initializer_list](#) of characters.

Parameters

l The [initializer_list](#) of characters to assign.

Returns

Reference to this string.

Definition at line 1148 of file basic_string.h.

Referenced by std::basic_string< char >::assign().

5.376.3.15 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::assign (const basic_string< _CharT,
_Traits, _Alloc > & __str)`

Set value to contents of another string.

Parameters

str Source string to use.

Returns

Reference to this string.

Definition at line 243 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::get_allocator()`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

5.376.3.16 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::at (
size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws [`out_of_range`](#) if the check fails.

Definition at line 847 of file `basic_string.h`.

5.376.3.17 `template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::at (
size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws [out_of_range](#) if the check fails. Success results in unsharing the string.

Definition at line 900 of file basic_string.h.

5.376.3.18 `template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back ()
[inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 876 of file basic_string.h.

5.376.3.19 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back
() const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 884 of file basic_string.h.

5.376.3.20 `template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ()
[inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 591 of file basic_string.h.

Referenced by `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

5.376.3.21 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 602 of file basic_string.h.

5.376.3.22 `template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str (
) const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1757 of file basic_string.h.

Referenced by `std::collate< _CharT >::do_compare()`, `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::collate< _CharT >::do_transform()`, and `std::basic_filebuf< char_type, traits_type >::open()`.

5.376.3.23 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity (
) const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 759 of file basic_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

5.376.3.24 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin (
) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 666 of file basic_string.h.

5.376.3.25 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend (
) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 674 of file basic_string.h.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2047

5.376.3.26 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::clear ()
[inline]`

Erases the string, making it empty.

Definition at line 786 of file basic_string.h.

Referenced by std::basic_stringbuf<_CharT, _Traits, _Alloc>::setbuf().

5.376.3.27 `template<typename _CharT, typename _Traits, typename _Alloc
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (
size_type __pos, size_type __n1, const _CharT * __s, size_type
__n2) const`

Compare substring against a character array.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

s character array to compare against.

n2 Number of characters of *s*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer < 0 if this substring is ordered before the string from *s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, '\0' has no special meaning.

Definition at line 980 of file basic_string.tcc.

References std::min().

5.376.3.28 `template<typename _CharT, typename _Traits, typename _Alloc
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (
size_type __pos, size_type __n, const basic_string<_CharT,
_Traits, _Alloc> & __str) const`

Compare substring to a string.

Parameters

pos Index of first character of substring.

n Number of characters in substring.

str String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *str.size()*. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 916 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::basic_string<_CharT, _Traits, _Alloc >::data()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

5.376.3.29 `template<typename _CharT, typename _Traits, typename _Alloc
> int std::basic_string<_CharT, _Traits, _Alloc >::compare (
size_type __pos1, size_type __n1, const basic_string<_CharT,
_Traits, _Alloc > & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

str String to compare against.

pos2 Index of first character of substring of *str*.

n2 Number of characters in substring of *str*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form the substring of *str* from the *n2* characters starting at *pos2*. Returns an integer < 0 if this substring is ordered before the substring of *str*, 0 if their values are

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2049

equivalent, or > 0 if this substring is ordered after the substring of *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 931 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::min()`.

5.376.3.30 `template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
const basic_string< _CharT, _Traits, _Alloc > & __str) const
[inline]`

Compare to a string.

Parameters

str String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before *str*, 0 if their values are equivalent, or > 0 if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2164 of file `basic_string.h`.

Referenced by `std::sub_match< _Bi_iter >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

5.376.3.31 `template<typename _CharT, typename _Traits , typename _Alloc
> int std::basic_string< _CharT, _Traits, _Alloc >::compare (
size_type __pos, size_type __n1, const _CharT * __s) const`

Compare substring to a C string.

Parameters

pos Index of first character of substring.

nl Number of characters in substring.

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *nl* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 964 of file `basic_string.tcc`.

References `std::min()`.

5.376.3.32 `template<typename _CharT, typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (const
_CharT * __s) const`

Compare to a C string.

Parameters

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 949 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, `std::min()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference

5.376.3.33 `template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::copy (_CharT * __s,
size_type __n, size_type __pos = 0) const`

Copy substring into C string.

Parameters

- s* C string to copy value into.
- n* Number of characters to copy.
- pos* Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

[*std::out_of_range*](#) If *pos* > *size()*.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than *size()*, *out_of_range* is thrown.

Definition at line 723 of file `basic_string.tcc`.

5.376.3.34 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc
>::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 683 of file `basic_string.h`.

5.376.3.35 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc
>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 692 of file `basic_string.h`.

5.376.3.36 `template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::data (
) const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1767 of file basic_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::collate< _CharT >::do_compare()`, `std::collate< _CharT >::do_transform()`, `std::basic_string< char >::find()`, `std::basic_string< char >::find_first_not_of()`, `std::basic_string< char >::find_first_of()`, `std::basic_string< char >::find_last_not_of()`, `std::basic_string< char >::find_last_of()`, `std::basic_string< char >::rfind()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, and `std::regex_traits< _Ch_type >::transform()`.

5.376.3.37 `template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty () const
[inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 794 of file basic_string.h.

Referenced by `std::operator>>()`.

5.376.3.38 `template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ()
[inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 610 of file basic_string.h.

Referenced by `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

5.376.3.39 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end (
) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 621 of file basic_string.h.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2053

5.376.3.40 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::erase (
size_type __pos = 0, size_type __n = npos) [inline]`

Remove characters.

Parameters

- pos* Index of first character to remove (default 0).
n Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are < *n* characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1338 of file basic_string.h.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_string< _CharT, _Traits, _Alloc >::resize()`.

5.376.3.41 `template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
iterator __position) [inline]`

Remove one character.

Parameters

- position* Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1354 of file basic_string.h.

5.376.3.42 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::iterator
std::basic_string< _CharT, _Traits, _Alloc >::erase (iterator
__first, iterator __last)`

Remove a range of characters.

Parameters

first Iterator referencing the first character to remove.

last Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 391 of file basic_string.tcc.

5.376.3.43 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find (const _CharT *
__s, size_type __pos, size_type __n) const`

Find position of a C substring.

Parameters

s C string to locate.

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 737 of file basic_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::find()`, and `std::basic_string< _CharT, _Traits, _Alloc >::find_first_of()`.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2055

5.376.3.44 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (const
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos =
0) const [inline]`

Find position of a string.

Parameters

str String to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1802 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find()`.

5.376.3.45 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (const
_CharT * __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

s C string to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1816 of file `basic_string.h`.

5.376.3.46 `template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::find (_CharT __c,
size_type __pos = 0) const`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 760 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::find()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.376.3.47 `template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of(
_CharT __c, size_type __pos = 0) const`

Find position of a different character.

Parameters

c Character to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 864 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.376.3.48 `template<typename _CharT, typename _Traits, typename
_Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_not_of(const basic_string<_CharT, _Traits, _Alloc
> & __str, size_type __pos = 0) const [inline]`

Find position of a character not in string.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2057

Parameters

str String containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2026 of file basic_string.h.

Referenced by std::basic_string<char>::find_first_not_of().

```
5.376.3.49 template<typename _CharT, typename _Traits, typename
        _Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
        std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
        const _CharT* __s, size_type __pos, size_type __n ) const
```

Find position of a character not in C substring.

Parameters

s C string containing characters to avoid.

pos Index of character to search from.

n Number of characters from *s* to consider.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 852 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc>::size().

```
5.376.3.50 template<typename _CharT, typename _Traits, typename
        _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>
        >::find_first_not_of ( const _CharT* __s, size_type __pos = 0 )
        const [inline]
```

Find position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2055 of file basic_string.h.

```
5.376.3.51  template<typename _CharT, typename _Traits, typename _Alloc>
              size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of
              ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
                __pos = 0 ) const  [inline]
```

Find position of a character of string.

Parameters

str String containing characters to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1904 of file basic_string.h.

Referenced by std::basic_string< char >::find_first_of().

```
5.376.3.52  template<typename _CharT, typename _Traits , typename
              _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
              std::basic_string< _CharT, _Traits, _Alloc >::find_first_of ( const
              _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character of C substring.

Parameters

s String containing characters to locate.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2059

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 816 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::find()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.376.3.53 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of
(const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

s String containing characters to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1932 of file `basic_string.h`.

5.376.3.54 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of
(_CharT __c, size_type __pos = 0) const [inline]`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1951 of file `basic_string.h`.

```
5.376.3.55 template<typename _CharT, typename _Traits, typename
               _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
               std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
               const _CharT* __s, size_type __pos, size_type __n ) const
```

Find last position of a character not in C substring.

Parameters

s C string containing characters to avoid.

pos Index of character to search back from.

n Number of characters from *s* to consider.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 875 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
5.376.3.56 template<typename _CharT, typename _Traits, typename
               _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
               >::find_last_not_of ( const _CharT* __s, size_type __pos = npow )
               const [inline]
```

Find last position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search back from (default end).

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2061

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2114 of file `basic_string.h`.

```
5.376.3.57  template<typename _CharT, typename _Traits, typename  
              _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type  
              std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (  
                _CharT __c, size_type __pos = npos ) const
```

Find last position of a different character.

Parameters

c Character to avoid.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 896 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
5.376.3.58  template<typename _CharT, typename _Traits, typename  
              _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc  
              >::find_last_not_of ( const basic_string< _CharT, _Traits, _Alloc >  
              & __str, size_type __pos = npos ) const  [inline]
```

Find last position of a character not in string.

Parameters

str String containing characters to avoid.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2085 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

5.376.3.59 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of
(_CharT __c, size_type __pos = npow) const [inline]`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 2012 of file `basic_string.h`.

5.376.3.60 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of
(const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
__pos = npow) const [inline]`

Find last position of a character of string.

Parameters

str String containing characters to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

Definition at line 1965 of file basic_string.h.

Referenced by std::basic_string< char >::find_last_of().

5.376.3.61 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (const
_CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character of C substring.

Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 831 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

5.376.3.62 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of
(const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a character of C string.

Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1993 of file basic_string.h.

5.376.3.63 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::front
() const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 868 of file basic_string.h.

5.376.3.64 `template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front ()
[inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 860 of file basic_string.h.

5.376.3.65 `template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc
>::get_allocator () const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1774 of file basic_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

5.376.3.66 `template<typename _CharT, typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::insert (size_type __pos, const _CharT
* __s, size_type __n)`

Insert a C substring.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

n The number of characters to insert.

Returns

Reference to this string.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2065

Exceptions

std::length_error If new length exceeds `max_size()`.

std::out_of_range If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 360 of file `basic_string.tcc`.

```
5.376.3.67  template<typename _CharT, typename _Traits, typename _Alloc>
            iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p, _CharT __c ) [inline]
```

Insert one character.

Parameters

p Iterator referencing position in string to insert at.

c The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts character *c* at position referenced by *p*. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1314 of file `basic_string.h`.

```
5.376.3.68  template<typename _CharT, typename _Traits, typename _Alloc>
            void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator
            __p, size_type __n, _CharT __c ) [inline]
```

Insert multiple characters.

Parameters

p Iterator referencing location in string to insert at.

n Number of characters to insert

c The character to insert.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts *n* copies of character *c* starting at the position referenced by iterator *p*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1165 of file `basic_string.h`.

```
5.376.3.69  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
                size_type __pos1, const basic_string< _CharT, _Traits, _Alloc > &
                __str ) [inline]
```

Insert value of a string.

Parameters

pos1 Iterator referencing location in string to insert at.

str The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts value of *str* starting at *pos1*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1211 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

```
5.376.3.70  template<typename _CharT, typename _Traits, typename _Alloc>
            void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator
                __p, initializer_list< _CharT > __l ) [inline]
```

Insert an `initializer_list` of characters.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2067

Parameters

- p* Iterator referencing location in string to insert at.
- l* The [initializer_list](#) of characters to insert.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Definition at line 1192 of file `basic_string.h`.

```
5.376.3.71  template<typename _CharT, typename _Traits, typename _Alloc>
            template<class _InputIterator > void std::basic_string< _CharT,
            _Traits, _Alloc >::insert ( iterator __p, _InputIterator __beg,
            _InputIterator __end ) [inline]
```

Insert a range of characters.

Parameters

- p* Iterator referencing location in string to insert at.
- beg* Start of range.
- end* End of range.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, [length_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1181 of file `basic_string.h`.

```
5.376.3.72  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos, size_type __n, _CharT __c ) [inline]
```

Insert multiple characters.

Parameters

- pos* Index in string to insert at.
- n* Number of characters to insert
- c* The character to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts *n* copies of character *c* starting at index *pos*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. If *pos* > `length()`, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1297 of file `basic_string.h`.

5.376.3.73 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
size_type __pos1, const basic_string< _CharT, _Traits, _Alloc > &
__str, size_type __pos2, size_type __n) [inline]`

Insert a substring.

Parameters

pos1 Iterator referencing location in string to insert at.

str The string to insert.

pos2 Start of characters in *str* to insert.

n Number of characters to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos1* > `size()` or *pos2* > *str.size()*.

Starting at *pos1*, insert *n* character of *str* beginning with *pos2*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. If *pos1* is beyond the end of this string or *pos2* is beyond the end of *str*, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2069

5.376.3.74 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
size_type __pos, const _CharT * __s) [inline]`

Insert a C string.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. If *pos* is beyond `end()`, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1274 of file `basic_string.h`.

5.376.3.75 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::length ()
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 707 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::collate< _CharT >::do_compare()`, `std::collate< _CharT >::do_transform()`, and `std::regex_traits< _Ch_type >::length()`.

5.376.3.76 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size () const [inline]`

Returns the [*size\(\)*](#) of the largest possible string.

Definition at line 712 of file `basic_string.h`.

Referenced by `std::getline()`, and `std::operator>>()`.

```
5.376.3.77  template<typename _CharT, typename _Traits, typename _Alloc>
             basic_string& std::basic_string< _CharT, _Traits, _Alloc
             >::operator+=( const basic_string< _CharT, _Traits, _Alloc > &
             __str ) [inline]
```

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

Definition at line 915 of file basic_string.h.

```
5.376.3.78  template<typename _CharT, typename _Traits, typename _Alloc>
             basic_string& std::basic_string< _CharT, _Traits, _Alloc
             >::operator+=( const _CharT * __s ) [inline]
```

Append a C string.

Parameters

s The C string to append.

Returns

Reference to this string.

Definition at line 924 of file basic_string.h.

```
5.376.3.79  template<typename _CharT, typename _Traits, typename _Alloc>
             basic_string& std::basic_string< _CharT, _Traits, _Alloc
             >::operator+=( _CharT __c ) [inline]
```

Append a character.

Parameters

c The character to append.

Returns

Reference to this string.

Definition at line 933 of file basic_string.h.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2071

5.376.3.80 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc
>::operator+=(initializer_list<_CharT> __l) [inline]`

Append an [initializer_list](#) of characters.

Parameters

l The [initializer_list](#) of characters to be appended.

Returns

Reference to this string.

Definition at line 946 of file `basic_string.h`.

5.376.3.81 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc
>::operator=(basic_string<_CharT, _Traits, _Alloc> && __str
) [inline]`

Move assign the value of *str* to this string.

Parameters

str Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 566 of file `basic_string.h`.

5.376.3.82 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc
>::operator=(const basic_string<_CharT, _Traits, _Alloc> &
__str) [inline]`

Assign the value of *str* to this string.

Parameters

str Source string.

Definition at line 532 of file `basic_string.h`.

5.376.3.83 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator= (const _CharT * __s) [inline]`

Copy contents of *s* into this string.

Parameters

s Source null-terminated string.

Definition at line 540 of file `basic_string.h`.

5.376.3.84 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator= (initializer_list< _CharT > __l) [inline]`

Set value to string constructed from initializer list.

Parameters

l [std::initializer_list](#).

Definition at line 578 of file `basic_string.h`.

5.376.3.85 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator= (_CharT __c) [inline]`

Set value to string of length 1.

Parameters

c Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 551 of file `basic_string.h`.

5.376.3.86 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc
>::operator[] (size_type __pos) const [inline]`

Subscript access to the data contained in the string.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2073

Parameters

pos The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 809 of file basic_string.h.

```
5.376.3.87  template<typename _CharT, typename _Traits, typename _Alloc>
             reference std::basic_string<_CharT, _Traits, _Alloc>::operator[] (
             size_type __pos ) [inline]
```

Subscript access to the data contained in the string.

Parameters

pos The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).) Unshares the string.

Definition at line 826 of file basic_string.h.

```
5.376.3.88  template<typename _CharT, typename _Traits, typename _Alloc>
             void std::basic_string<_CharT, _Traits, _Alloc>::push_back (
             _CharT __c ) [inline]
```

Append a single character.

Parameters

c Character to append.

Definition at line 1034 of file basic_string.h.

Referenced by std::collate<_CharT>::do_transform(), and std::operator>>().

5.376.3.89 `template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc
>::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 630 of file `basic_string.h`.

5.376.3.90 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc
>::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 639 of file `basic_string.h`.

5.376.3.91 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc
>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 657 of file `basic_string.h`.

5.376.3.92 `template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend
() [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 648 of file `basic_string.h`.

5.376.3.93 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace
(size_type __pos, size_type __n, const basic_string< _CharT,
_Traits, _Alloc > & __str) [inline]`

Replace characters with value from another string.

Parameters

pos Index of first character to replace.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2075

n Number of characters to be replaced.

str String to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*pos*,*pos*+*n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, [`out_of_range`](#) is thrown. If the length of the result exceeds `max_size()`, [`length_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1393 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

5.376.3.94 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace
(iterator __i1, iterator __i2, const basic_string< _CharT, _Traits,
_Alloc > & __str) [inline]`

Replace range of characters with string.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

str String value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*i1*,*i2*). In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, [`length_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1499 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

5.376.3.95 `template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, const _CharT * __s, size_type __n
) [inline]`

Replace range of characters with C substring.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.
- n* Number of characters from *s* to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1517 of file `basic_string.h`.

5.376.3.96 `template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, const _CharT * __s) [inline]`

Replace range of characters with C string.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2077

Removes the characters in the range [i1,i2). In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file `basic_string.h`.

```
5.376.3.97  template<typename _CharT, typename _Traits, typename _Alloc>
              basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace
              ( size_type __pos1, size_type __n1, const basic_string< _CharT,
                _Traits, _Alloc > & __str, size_type __pos2, size_type __n2 )
              [inline]
```

Replace characters with value from another string.

Parameters

pos1 Index of first character to replace.
n1 Number of characters to be replaced.
str String to insert.
pos2 Index of first character of *str* to use.
n2 Number of characters from *str* to use.

Returns

Reference to this string.

Exceptions

`std::out_of_range` If *pos1* > `size()` or *pos2* > `str.size()`.
`std::length_error` If new length exceeds `max_size()`.

Removes the characters in the range [*pos1*,*pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1415 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

```
5.376.3.98  template<typename _CharT, typename _Traits, typename _Alloc>
              basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace
              ( size_type __pos, size_type __n1, size_type __n2, _CharT __c )
              [inline]
```

Replace characters with multiple characters.

Parameters

pos Index of first character to replace.
n1 Number of characters to be replaced.
n2 Number of characters to insert.
c Character to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* > *size()*.
[*std::length_error*](#) If new length exceeds *max_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, [*out_of_range*](#) is thrown. If the length of result exceeds *max_size()*, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1481 of file *basic_string.h*.

5.376.3.99 `template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, size_type __n, _CharT __c)
 [inline]`

Replace range of characters with multiple characters.

Parameters

i1 Iterator referencing start of range to replace.
i2 Iterator referencing end of range to replace.
n Number of characters to insert.
c Character to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds *max_size()*.

Removes the characters in the range [*i1*,*i2*). In place, *n* copies of *c* are inserted. If the length of result exceeds *max_size()*, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1559 of file *basic_string.h*.

5.376 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2079

5.376.3.100 `template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator > basic_string& std::basic_string<
_CharT, _Traits, _Alloc>::replace (iterator __i1, iterator __i2,
_InputIterator __k1, _InputIterator __k2) [inline]`

Replace range of characters with range.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

k1 Iterator referencing start of range to insert.

k2 Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds `max_size()`, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1582 of file `basic_string.h`.

5.376.3.101 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<
_CharT, _Traits, _Alloc>::replace (size_type __pos, size_type
__n1, const _CharT * __s, size_type __n2)`

Replace characters with value of a C substring.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

n2 Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos1* > *size()*.

std::length_error If new length exceeds *max_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, the first *n2* characters of *s* are inserted, or all of *s* if *n2* is too large. If *pos* is beyond end of string, *out_of_range* is thrown. If the length of result exceeds *max_size()*, *length_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 414 of file *basic_string.tcc*.

5.376.3.102 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::replace (size_type __pos, size_type __n1, const _CharT * __s
) [inline]`

Replace characters with value of a C string.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos* > *size()*.

std::length_error If new length exceeds *max_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, the characters of *s* are inserted. If *pos* is beyond end of string, *out_of_range* is thrown. If the length of result exceeds *max_size()*, *length_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1458 of file *basic_string.h*.

5.376.3.103 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::replace (iterator __i1, iterator __i2, initializer_list< _CharT
> __l) [inline]`

Replace range of characters with *initializer_list*.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2081

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- l* The [initializer_list](#) of characters to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, [length_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1650 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

5.376.3.104 `template<typename _CharT , typename _Traits , typename _Alloc
> void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
size_type __res_arg = 0)`

Attempt to preallocate enough memory for specified number of characters.

Parameters

res_arg Number of characters required.

Exceptions

[std::length_error](#) If *res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, [length_error](#) is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 502 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::num_get<_CharT, _InIter>::do_get()`, and `std::operator>>()`.

5.376.3.105 `template<typename _CharT, typename _Traits, typename _Alloc>
> void std::basic_string<_CharT, _Traits, _Alloc>::resize (
size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

Definition at line 640 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::erase()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`.

5.376.3.106 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
size_type __n) [inline]`

Resizes the string to the specified number of characters.

Parameters

- n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 739 of file `basic_string.h`.

Referenced by `std::basic_string<char>::resize()`.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2083

5.376.3.107 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
__pos = npos) const [inline]`

Find last position of a string.

Parameters

str String to locate.

pos Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1846 of file `basic_string.h`.

Referenced by `std::basic_string< char >::rfind()`.

5.376.3.108 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::rfind (_CharT __c,
size_type __pos = npos) const`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 799 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.376.3.109 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a C string.

Parameters

s C string to locate.

pos Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1874 of file `basic_string.h`.

5.376.3.110 `template<typename _CharT, typename _Traits , typename
_Alloc > basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::rfind (const
_CharT * __s, size_type __pos, size_type __n) const`

Find last position of a C substring.

Parameters

s C string to locate.

pos Index of character to search back from.

n Number of characters from *s* to search for.

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 778 of file `basic_string.tcc`.

References `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.376 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2085

5.376.3.111 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit ()
[inline]`

A non-binding request to reduce [capacity\(\)](#) to [size\(\)](#).

Definition at line 745 of file `basic_string.h`.

5.376.3.112 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size ()
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 701 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::bitset< _S_match_flag_last >::bitset()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::find()`, `std::basic_string< char >::find()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of()`, `std::basic_string< char >::find_first_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_of()`, `std::basic_string< char >::find_first_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of()`, `std::basic_string< char >::find_last_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_last_of()`, `std::basic_string< char >::find_last_of()`, `std::basic_string< char >::insert()`, `std::operator+`, `std::basic_string< char >::replace()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, `std::basic_string< _CharT, _Traits, _Alloc >::rfind()`, `std::basic_string< char >::rfind()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, and `std::regex_traits< _Ch_type >::transform()`.

5.376.3.113 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr ()
size_type __pos = 0, size_type __n = npos) const [inline]`

Get a substring.

Parameters

pos Index of first character (default 0).

n Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

[*std::out_of_range*](#) If `pos > size()`.

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, [*out_of_range*](#) is thrown.

Definition at line 2146 of file `basic_string.h`.

5.376.3.114 `template<typename _CharT, typename _Traits, typename _Alloc
> void std::basic_string< _CharT, _Traits, _Alloc >::swap (
basic_string< _CharT, _Traits, _Alloc > & __s)`

Swap contents with another string.

Parameters

s String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

Definition at line 519 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`.

Referenced by `std::swap()`.

5.376.4 Member Data Documentation

5.376.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> const basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 271 of file `basic_string.h`.

The documentation for this class was generated from the following files:

- [basic_string.h](#)
- [basic_string.tcc](#)

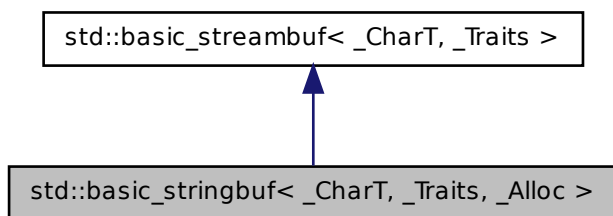
5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

The actual work of input and output (for `std::string`).

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

Inheritance diagram for `std::basic_stringbuf< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `__string_type::size_type` **__size_type**
- typedef `basic_streambuf< char_type, traits_type >` **__streambuf_type**
- typedef `basic_string< char_type, _Traits, _Alloc >` **__string_type**
- typedef `_Alloc` **allocator_type**
- typedef `_CharT` **char_type**
- typedef `traits_type::int_type` **int_type**
- typedef `traits_type::off_type` **off_type**
- typedef `traits_type::pos_type` **pos_type**
- typedef `_Traits` **traits_type**

Public Member Functions

- `basic_stringbuf (ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `basic_stringbuf (const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize in_avail ()`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`

- [int_type sputbackc](#) ([char_type __c](#))
- [int_type sputc](#) ([char_type __c](#))
- [streamsize sputn](#) ([const char_type *__s](#), [streamsize __n](#))
- [void stoss](#) ()
- [void str](#) ([const __string_type &__s](#))
- [__string_type str](#) () [const](#)
- [int_type sungetc](#) ()

Protected Member Functions

- [void _M_stringbuf_init](#) ([ios_base::openmode __mode](#))
 - [void _M_sync](#) ([char_type *__base](#), [__size_type __i](#), [__size_type __o](#))
 - [void _M_update_egptr](#) ()
 - [void gbump](#) ([int __n](#))
 - [virtual void imbue](#) ([const locale &](#))
 - [virtual int_type overflow](#) ([int_type __c=traits_type::eof\(\)](#))
 - [virtual int_type overflow](#) ([int_type=traits_type::eof\(\)](#))
 - [virtual int_type pbackfail](#) ([int_type __c=traits_type::eof\(\)](#))
 - [virtual int_type pbackfail](#) ([int_type=traits_type::eof\(\)](#))
 - [void pbump](#) ([int __n](#))
 - [virtual pos_type seekoff](#) ([off_type](#), [ios_base::seekdir](#), [ios_base::openmode=ios_base::in|ios_base::out](#))
 - [virtual pos_type seekoff](#) ([off_type __off](#), [ios_base::seekdir __way](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
 - [virtual pos_type seekpos](#) ([pos_type __sp](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
 - [virtual pos_type seekpos](#) ([pos_type](#), [ios_base::openmode=ios_base::in|ios_base::out](#))
 - [virtual basic_streambuf< char_type, _Traits > * setbuf](#) ([char_type *](#), [streamsize](#))
 - [virtual __streambuf_type * setbuf](#) ([char_type *__s](#), [streamsize __n](#))
 - [void setg](#) ([char_type *__gbeg](#), [char_type *__gnext](#), [char_type *__gend](#))
 - [void setp](#) ([char_type *__pbeg](#), [char_type *__pend](#))
 - [virtual streamsize showmanyc](#) ()
 - [virtual int sync](#) ()
 - [virtual int_type uflow](#) ()
 - [virtual int_type underflow](#) ()
 - [virtual streamsize xsgetn](#) ([char_type *__s](#), [streamsize __n](#))
 - [virtual streamsize xsputn](#) ([const char_type *__s](#), [streamsize __n](#))
-
- [char_type * eback](#) () [const](#)
 - [char_type * gptr](#) () [const](#)
 - [char_type * egptr](#) () [const](#)
-
- [char_type * pbase](#) () [const](#)
 - [char_type * pptr](#) () [const](#)
 - [char_type * ep_ptr](#) () [const](#)

Protected Attributes

- [ios_base::openmode](#) [_M_mode](#)
- [__string_type](#) [_M_string](#)

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
 - [streamsize](#) [__copy_streambufs_eof](#) ([__streambuf_type](#) *, [__streambuf_type](#) *,
bool &)
 - class [basic_ios](#)< [char_type](#), [traits_type](#) >
 - class [basic_istream](#)< [char_type](#), [traits_type](#) >
 - class [basic_ostream](#)< [char_type](#), [traits_type](#) >
 - `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _-`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
 - class [istreambuf_iterator](#)< [char_type](#), [traits_type](#) >
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
 - `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
 - class [ostreambuf_iterator](#)< [char_type](#), [traits_type](#) >
-
- [char_type](#) * [_M_in_beg](#)
 - [char_type](#) * [_M_in_cur](#)
 - [char_type](#) * [_M_in_end](#)
 - [char_type](#) * [_M_out_beg](#)
 - [char_type](#) * [_M_out_cur](#)
 - [char_type](#) * [_M_out_end](#)
 - [locale](#) [_M_buf_locale](#)
 - [locale](#) [pubimbue](#) (const [locale](#) & __loc)
 - [locale](#) [getloc](#) () const
 - [__streambuf_type](#) * [pubsetbuf](#) ([char_type](#) * __s, [streamsize](#) __n)

- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()

5.377.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_stringbuf< _CharT, _Traits, _Alloc >`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.). For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 58 of file `sstream`.

5.377.2 Member Typedef Documentation

5.377.2.1 `template<typename _CharT, typename _Traits, typename _Alloc> typedef basic_streambuf<char_type, traits_type> std::basic_stringbuf< _CharT, _Traits, _Alloc >::__streambuf_type`

This is a non-standard type.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 71 of file `sstream`.

5.377.2.2 `template<typename _CharT, typename _Traits, typename _Alloc> typedef _CharT std::basic_stringbuf< _CharT, _Traits, _Alloc >::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 62 of file `sstream`.

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

5.377.2.3 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::int_type std::basic_stringbuf< _CharT, _Traits,
_Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 67 of file sstream.

5.377.2.4 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::off_type std::basic_stringbuf< _CharT, _Traits,
_Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 69 of file sstream.

5.377.2.5 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::pos_type std::basic_stringbuf< _CharT, _Traits,
_Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 68 of file sstream.

5.377.2.6 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _Traits std::basic_stringbuf< _CharT, _Traits, _Alloc
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 63 of file sstream.

5.377.3 Constructor & Destructor Documentation

5.377.3.1 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf
(ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, explicit]`

Starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

Definition at line 92 of file sstream.

5.377.3.2 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf
(const __string_type & __str, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, explicit]`

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

Definition at line 105 of file sstream.

5.377.4 Member Function Documentation

5.377.4.1 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eback () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.377.4.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::egptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.377.4.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::epptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by std::basic_streambuf< _CharT, _Traits >::xsputn().

5.377.4.4 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.377.4.5 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.377.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

5.377.4.7 `template<typename _CharT, typename _Traits> virtual void
std::basic_streambuf< _CharT, _Traits >::imbue (const locale &)
[inline, protected, virtual, inherited]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 554 of file streambuf.

5.377.4.8 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::in_avail () [inline,
inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file streambuf.

5.377.4.9 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::overflow (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

eof() to indicate failure, something else (usually *c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not eof().

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented in [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#).

Definition at line 746 of file streambuf.

Referenced by [std::basic_streambuf<_CharT, _Traits>::xsputn\(\)](#).

5.377.4.10 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

Note

Base class version does nothing, returns eof().

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 702 of file streambuf.

5.377.4.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

5.377.4.12 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.377.4.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.377.4.14 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

5.377.4.15 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file streambuf.

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

5.377.4.16 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.377.4.17 `template<typename _CharT, typename _Traits>
__streambuf_type* std::basic_streambuf< _CharT, _Traits
>::pubsetbuf (char_type * __s, streamsize __n) [inline,
inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.377.4.18 `template<typename _CharT, typename _Traits> int
std::basic_streambuf< _CharT, _Traits >::pubsync ()
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

5.377.4.19 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

5.377.4.20 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file streambuf.

5.377.4.21 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Definition at line 592 of file streambuf.

5.377.4.22 `template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits,
_Alloc >::setbuf(char_type * __s, streamsize __n) [inline,
protected, virtual]`

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

this

If no buffer has already been created, and both *s* and *n* are non-zero, then *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 196 of file sstream.

References `std::basic_string< _CharT, _Traits, _Alloc >::clear()`.

5.377.4.23 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT,_Traits >::setbuf(char_type *, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns *this*.

Definition at line 569 of file streambuf.

5.377.4.24 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setg (char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

5.377.4.25 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setp (char_type
* __pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.
pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.377.4.26 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sgetc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

5.377 std::basic_stringbuf<_CharT, _Traits, _Alloc> Class Template Reference

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.377.4.27 `template<typename _CharT, typename _Traits> streamsize
std::basic_stringbuf<_CharT, _Traits>::sgetn(char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file streambuf.

5.377.4.28 `template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf<_CharT, _Traits, _Alloc>
::showmanyc() [inline, protected, virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 164 of file sstream.

5.377.4.29 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::snextc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls [sbumpc\(\)](#), and if that function returns `traits::eof()`, so does this function. Otherwise, [sgetc\(\)](#).

Definition at line 276 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.377.4.30 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc(char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

5.377.4.31 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc(char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

5.377.4.32 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sputn (const char_type
* __s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.377.4.33 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::stossc () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.377.4.34 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (const
__string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Deallocates any previous stored sequence, then copies *s* to use as a new one.

Definition at line 144 of file sstream.

References `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.377.4.35 `template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
) const [inline]`

Copying out the string buffer.

Returns

A copy of one of the underlying sequences.

If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1

Definition at line 120 of file sstream.

5.377.4.36 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Definition at line 375 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

5.377.4.37 `template<typename _CharT, typename _Traits> virtual int
std::basic_streambuf< _CharT, _Traits >::sync (void)
[inline, protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 605 of file streambuf.

5.377.4.38 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 678 of file streambuf.

5.377.4.39 `template<class _CharT, class _Traits, class _Alloc >
 basic_stringbuf< _CharT, _Traits, _Alloc >::int_type
 std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow ()
 [protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 131 of file `sstream.tcc`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::M_mode`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::in`.

5.377.4.40 `template<typename _CharT, typename _Traits > streamsize
 std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type *
 __s, streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

5.377 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

```
5.377.4.41  template<typename _CharT, typename _Traits> streamsize
              std::basic_streambuf<_CharT, _Traits>::xsputn ( const
              char_type * __s, streamsize __n ) [protected, virtual,
              inherited]
```

Multiple character insertion.

Parameters

`s` A buffer area.

`n` Maximum number of characters to write.

Returns

The number of characters written.

Writes `s[0]` through `s[n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.377.5 Member Data Documentation

5.377.5.1 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::_M_buf_locale
[protected, inherited]`

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`.

5.377.5.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_beg
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file streambuf.

5.377.5.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file streambuf.

5.377.5.4 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

5.377 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 182 of file streambuf.

5.377.5.5 `template<typename _CharT, typename _Traits, typename _Alloc>
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 77 of file sstream.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow().

5.377.5.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 183 of file streambuf.

5.377.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.377.5.8 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following files:

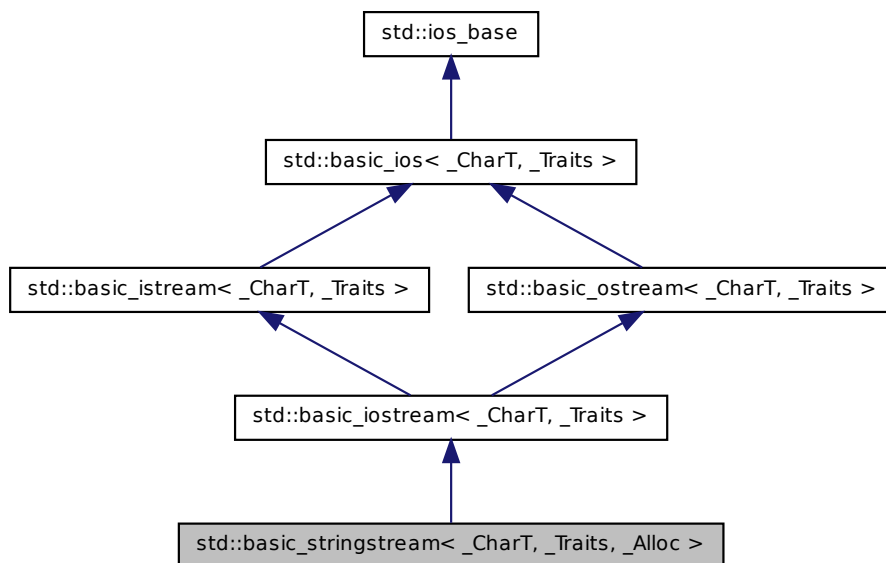
- [sstream](#)
- [sstream.tcc](#)

5.378 `std::basic_stringstream< _CharT, _Traits, _- Alloc >` Class Template Reference

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_iostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_stringstream< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`

- typedef [basic_stringbuf](#)< [_CharT](#), [_Traits](#), [_Alloc](#) > [__stringbuf_type](#)
- typedef [_Alloc](#) [allocator_type](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_CharT](#) [char_type](#)
- enum [event](#) { [erase_event](#), [imbue_event](#), [copyfmt_event](#) }
- typedef void(* [event_callback](#))(event, [ios_base](#) &, int)
- typedef [_Ios_Fmtflags](#) [fmtflags](#)
- typedef [_Traits::int_type](#) [int_type](#)
- typedef [traits_type::int_type](#) [int_type](#)
- typedef int [io_state](#)
- typedef [_Ios_Iostate](#) [iostate](#)
- typedef [traits_type::off_type](#) [off_type](#)
- typedef [_Traits::off_type](#) [off_type](#)
- typedef int [open_mode](#)
- typedef [_Ios_Openmode](#) [openmode](#)
- typedef [traits_type::pos_type](#) [pos_type](#)
- typedef [_Traits::pos_type](#) [pos_type](#)
- typedef int [seek_dir](#)
- typedef [_Ios_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [_Traits](#) [traits_type](#)

- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)

Public Member Functions

- [basic_stringstream](#) ([ios_base::openmode](#) __m=[ios_base::out](#)|[ios_base::in](#))
- [basic_stringstream](#) (const [__string_type](#) &__str, [ios_base::openmode](#) __m=[ios_base::out](#)|[ios_base::in](#))
- [~basic_stringstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const

- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `__ostream_type & flush ()`
- `streamsize gcount () const`
- `template<>`
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type`
`__delim)`
- `template<>`
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type`
`__delim)`
- `locale getloc () const`
- `bool good () const`
- `template<>`
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `template<>`
`basic_istream< char > & ignore (streamsize __n)`
- `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale &__loc)`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `__stringbuf_type * rdbuf () const`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits`
`> *__sb)`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `__string_type str () const`
- `void str (const __string_type &__s)`
- `pos_type tellp ()`
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * _-`
`_tiestr)`

- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- void [unsetf](#) ([fmtflags](#) __mask)
- [char_type](#) [widen](#) (char __c) const
- [streamsize](#) [width](#) ([streamsize](#) __wide)
- [streamsize](#) [width](#) () const
- [__istream_type](#) & [operator>>](#) ([__istream_type](#) &(*__pf)([__istream_type](#) &))
- [__istream_type](#) & [operator>>](#) ([__ios_type](#) &(*__pf)([__ios_type](#) &))
- [__istream_type](#) & [operator>>](#) ([ios_base](#) &(*__pf)([ios_base](#) &))

Arithmetic Extractors

All the [operator>>](#) functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type [std::basic_istream::sentry](#) with the second argument ([noskipws](#)) set to false. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the [sentry](#) status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, [ios_base::badbit](#) will be turned on in the stream's error state without causing an [ios_base::failure](#) to be thrown. The original exception will then be rethrown.

- [__istream_type](#) & [operator>>](#) (bool &__n)
- [__istream_type](#) & [operator>>](#) (short &__n)
- [__istream_type](#) & [operator>>](#) (unsigned short &__n)
- [__istream_type](#) & [operator>>](#) (int &__n)
- [__istream_type](#) & [operator>>](#) (unsigned int &__n)
- [__istream_type](#) & [operator>>](#) (long &__n)
- [__istream_type](#) & [operator>>](#) (unsigned long &__n)
- [__istream_type](#) & [operator>>](#) (long long &__n)
- [__istream_type](#) & [operator>>](#) (unsigned long long &__n)
- [__istream_type](#) & [operator>>](#) (float &__f)
- [__istream_type](#) & [operator>>](#) (double &__f)
- [__istream_type](#) & [operator>>](#) (long double &__f)
- [__istream_type](#) & [operator>>](#) (void *&__p)
- [__istream_type](#) & [operator>>](#) ([__streambuf_type](#) *__sb)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type [std::basic_istream::sentry](#) with the second argument ([noskipws](#)) set to true. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the [sentry](#) status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by [`gcount\(\)`](#).

If an exception is thrown during extraction, [`ios_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`int_type get \(\)`](#)
- [`__istream_type & get \(char_type &__c\)`](#)
- [`__istream_type & get \(char_type *__s, streamsize __n, char_type __delim\)`](#)
- [`__istream_type & get \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & get \(__streambuf_type &__sb, char_type __delim\)`](#)
- [`__istream_type & get \(__streambuf_type &__sb\)`](#)
- [`__istream_type & getline \(char_type *__s, streamsize __n, char_type __delim\)`](#)
- [`__istream_type & getline \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & ignore \(\)`](#)
- [`__istream_type & ignore \(streamsize __n\)`](#)
- [`__istream_type & ignore \(streamsize __n, int_type __delim\)`](#)
- [`int_type peek \(\)`](#)
- [`__istream_type & read \(char_type *__s, streamsize __n\)`](#)
- [`streamsize readsome \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & putback \(char_type __c\)`](#)
- [`__istream_type & unget \(\)`](#)
- [`int sync \(\)`](#)
- [`pos_type tellg \(\)`](#)
- [`__istream_type & seekg \(pos_type\)`](#)
- [`__istream_type & seekg \(off_type, ios_base::seekdir\)`](#)
- [`operator void * \(\) const`](#)
- [`bool operator! \(\) const`](#)
- [`__ostream_type & operator<< \(__ostream_type &\(__pf\)\(__ostream_type &\)\)`](#)
- [`__ostream_type & operator<< \(__ios_type &\(__pf\)\(__ios_type &\)\)`](#)
- [`__ostream_type & operator<< \(ios_base &\(__pf\)\(ios_base &\)\)`](#)

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type [`std::basic_ostream::sentry`](#). This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, [`ios_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`__ostream_type & operator<< \(long __n\)`](#)

- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`

- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- template<typename _ValueT >
 [__istream_type](#) & [_M_extract](#) (_ValueT &__v)
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & [_M_insert](#) (_ValueT __v)
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

Friends

- class `sentry`
- class `sentry`

5.378.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_stringstream< _CharT, _Traits, _Alloc >
```

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 476 of file `sstream`.

5.378.2 Member Typedef Documentation

5.378.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_istream<_CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `istream`.

5.378.2.2 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ostream<_CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `ostream`.

5.378.2.3 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_istream<_CharT, _Traits>::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file `istream`.

5.378.2.4 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream<_CharT, _Traits>](#), [std::basic_ostream<char, _Traits>](#), and [std::basic_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

5.378.2.5 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file ostream.

5.378.2.6 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 59 of file istream.

5.378.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _CharT std::basic_stringstream<_CharT, _Traits, _Alloc>
>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 480 of file sstream.

5.378.2.8 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the [ios_base](#) object.

int The integer provided when the callback was registered.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2123

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 444 of file `ios_base.h`.

5.378.2.9 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.378.2.10 `template<typename _CharT, typename _Traits> typedef
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 60 of file istream.

5.378.2.11 `template<typename _CharT , typename _Traits , typename _Alloc
> typedef traits_type::int_type std::basic_stringstream< _CharT,
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 485 of file sstream.

5.378.2.12 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.378.2.13 `template<typename _CharT, typename _Traits, typename _Alloc
> typedef traits_type::off_type std::basic_stringstream<_CharT,
_Traits, _Alloc>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 487 of file `sstream`.

5.378.2.14 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 62 of file `istream`.

5.378.2.15 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

5.378.2.16 `template<typename _CharT, typename _Traits, typename _Alloc
> typedef traits_type::pos_type std::basic_stringstream< _CharT,
_Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 486 of file sstream.

5.378.2.17 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 61 of file istream.

5.378.2.18 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file ios_base.h.

5.378.2.19 `template<typename _CharT, typename _Traits> typedef
_Traits std::basic_istream< _CharT, _Traits >::traits_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2127

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istream<_CharT, _Traits, _Alloc>](#).

Definition at line 63 of file `istream`.

5.378.2.20 `template<typename _CharT, typename _Traits, typename _Alloc
> typedef _Traits std::basic_stringstream<_CharT, _Traits, _Alloc
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 481 of file `sstream`.

5.378.3 Member Enumeration Documentation

5.378.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

5.378.4 Constructor & Destructor Documentation

5.378.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc
>::basic_stringstream (ios_base::openmode __m =
ios_base::out | ios_base::in) [inline, explicit]`

Default constructor starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

Initializes `sb` using `mode`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 510 of file sstream.

```
5.378.4.2  template<typename _CharT , typename _Traits , typename
             _Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc
             >::basic_stringstream ( const __string_type & __str,
             ios_base::openmode __m = ios_base::out | ios_base::in )
             [inline, explicit]
```

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

Initializes *sb* using *str* and *mode*, and passes *&sb* to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 526 of file sstream.

```
5.378.4.3  template<typename _CharT , typename _Traits , typename
             _Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc
             >::~~basic_stringstream ( ) [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 537 of file sstream.

5.378.5 Member Function Documentation

```
5.378.5.1  const locale& std::ios_base::_M_getloc ( ) const [inline,
             inherited]
```

Locale access.

Returns

A reference to the current locale.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2129

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

5.378.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream<_CharT, _Traits>::_M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

5.378.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.378.5.4 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.378.5.5 `template<typename _CharT, typename _Traits> basic_ios<
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.378.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::eof () const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), and std::basic_istream< _CharT, _Traits >::unget().

5.378.5.7 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.378.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

```

#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file `basic_ios.h`.

5.378.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.378.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

5.378 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference 2133

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.378.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.378.5.12 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.378.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.378.5.14 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.378.5.15 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.378.5.16 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.17 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get (char_type & __c) [inherited]`

Simple extraction.

Parameters

`c` The character in which to store data.

Returns

`*this`

Tries to extract a character and store it in `c`. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.18 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

`s` Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.378.5.19 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

5.378.5.20 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::get (__streambuf_type &
__sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb, widen('\n'))`.

Definition at line 366 of file `istream`.

5.378.5.21 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get
(char_type * __s, streamsize __n, char_type __delim)
[inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.378.5.22 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim)`
[**inherited**]

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

delim A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2139

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.378.5.23 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

5.378.5.24 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.378.5.25 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.378.5.26 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(streamsize __n) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.378.5.27 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `delim` (in this case, the character is extracted); note that this condition will never occur if `delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.28 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.378.5.29 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc) [inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.378.5.30 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.378.5.31 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios_base.h.

5.378.5.32 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.
dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file basic_ios.h.

5.378.5.33 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits>::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file basic_ios.h.

5.378.5.34 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file basic_ios.h.

5.378.5.35 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.378.5.36 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.378.5.37 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.378.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 108 of file ostream.

5.378.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file ostream.

5.378.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 127 of file ostream.

5.378.5.41 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.378.5.42 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (short __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.378.5.43 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (int __n) [inherited]`

Basic arithmetic inserters.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2147

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.378.5.44 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file `ostream`.

5.378.5.45 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (const void * __p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.378.5.46 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.378.5.47 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.378.5.48 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.49 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.378.5.50 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.378.5.51 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.378.5.52 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (short & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.53 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

5.378.5.54 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

5.378.5.55 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (ios_base
&(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

5.378.5.56 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

5.378.5.57 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long double
& __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

5.378.5.58 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.378.5.59 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (long & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.378.5.60 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (void *& __p
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

5.378.5.61 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 194 of file istream.

5.378.5.62 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.63 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.64 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file istream.

5.378.5.65 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline,
inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 120 of file `istream`.

5.378.5.66 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 124 of file `istream`.

5.378.5.67 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.378.5.68 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
long long & __n) [inline, inherited]`

Basic arithmetic extractors.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2157

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

5.378.5.69 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek(void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.70 `streamsize std::ios_base::precision() const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.378.5.71 `streamsize std::ios_base::precision (streamsize __prec)`
`[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

5.378.5.72 `template<typename _CharT, typename _Traits > basic_ostream<`
`_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (`
`char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.378.5.73 `template<typename _CharT, typename _Traits > basic_istream<`
`_CharT, _Traits > & std::basic_istream< _CharT, _Traits`
`>::putback (char_type __c) [inherited]`

Unextracting a single character.

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2159

Parameters

c The character to push back into the input stream.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

5.378.5.74 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.378.5.75 `template<typename _CharT, typename _Traits, typename _Alloc
> __stringbuf_type* std::basic_stringstream< _CharT, _Traits,
_Alloc >::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current [basic_stringbuf](#) buffer.

This hides both signatures of [std::basic_ios::rdbuf\(\)](#).

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 548 of file sstream.

5.378.5.76 `template<typename _CharT, typename _Traits> basic_streambuf<
_CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
basic_streambuf< _CharT, _Traits > * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument [rdbuf\(\)](#), which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References [std::basic_ios< _CharT, _Traits >::clear\(\)](#).

5.378.5.77 `template<typename _CharT, typename _Traits> istate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.378.5.78 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.79 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rddbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.378.5.80 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.378.5.81 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg
(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.82 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg
(pos_type __pos) [inherited]`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.83 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.84 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

5.378 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2165

If `fail()` is not true, calls `rdbuf()` -> `pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.378.5.85 `fmtflags std::ios_base::setf(fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.378.5.86 `fmtflags std::ios_base::setf(fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.378.5.87 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.378.5.88 `template<typename _CharT , typename _Traits , typename _Alloc
> __string_type std::basic_stringstream< _CharT, _Traits, _Alloc
>::str () const [inline]`

Copying out the string buffer.

Returns

`rdbuf() -> str()`

Definition at line 556 of file `sstream`.

5.378.5.89 `template<typename _CharT , typename _Traits , typename _Alloc
> void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
const __string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Calls `rdbuf()` \rightarrow `str(s)`.

Definition at line 566 of file `sstream`.

5.378.5.90 `template<typename _CharT, typename _Traits> int
std::basic_istream< _CharT, _Traits>::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()` \rightarrow `pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits>::pubsync()`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

5.378.5.91 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.378.5.92 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg(void) [inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.378.5.93 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.378.5.94 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits>* __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file basic_ios.h.

5.378.5.95 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits
>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, [std::cin](#) is tied to [std::cout](#).

Definition at line 285 of file basic_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.378.5.96 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void) [inherited]`

Unextracting the previous character.

Returns

*this

If [rdbuf\(\)](#) is not null, calls `rdbuf()->sungetc(c)`.

If [rdbuf\(\)](#) is null or if `sungetc()` fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 744 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

5.378.5.97 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.378.5.98 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic_ios.h.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.378 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference 2171

5.378.5.99 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.378.5.100 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

5.378.5.101 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

`*this`

Characters are copied from *s* and inserted into the stream until one of the following happens:

- n characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed `char` and unsigned `char`.

5.378.5.102 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.378.6 Member Data Documentation

5.378.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.378.6.2 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

5.378 `std::basic_stringstream< _CharT, _Traits, _Alloc >` Class Template Reference

2173

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.378.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

5.378.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.378.6.5 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.378.6.6 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.378.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.378.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.378.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.378.6.10 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.378.6.11 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.378.6.12 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.378.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.378.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.378.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.378.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios_base.h.

5.378.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.378.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.378.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.378.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios_base.h.

5.378.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.378.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.378.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.378.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios_base.h.

5.378.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file ios_base.h.

5.378.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios_base.h.

5.378.6.27 `const fmtflags std::ios_base::showpoint` `[static, inherited]`

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file `ios_base.h`.

5.378.6.28 `const fmtflags std::ios_base::showpos` `[static, inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

5.378.6.29 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.378.6.30 `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.378.6.31 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.378.6.32 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

- [sstream](#)

5.379 std::bernoulli_distribution Class Reference

A Bernoulli random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef bool [result_type](#)

Public Member Functions

- [bernoulli_distribution](#) (double __p=0.5)
- **bernoulli_distribution** (const [param_type](#) &__p)
- [result_type](#) max () const
- [result_type](#) min () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator**() (_UniformRandomNumberGenerator &__urng, const
[param_type](#) &__p)
- double [p](#) () const
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

5.379.1 Detailed Description

A Bernoulli random number distribution. Generates a sequence of true and false values with likelihood p that true will come up and $(1 - p)$ that false will appear.

Definition at line 3240 of file random.h.

5.379.2 Member Typedef Documentation

5.379.2.1 typedef bool std::bernoulli_distribution::result_type

The type of the range of the distribution.

Definition at line 3244 of file random.h.

5.379.3 Constructor & Destructor Documentation

5.379.3.1 `std::bernoulli_distribution::bernoulli_distribution (double __p = 0.5) [inline, explicit]`

Constructs a Bernoulli distribution with likelihood `p`.

Parameters

`__p` [IN] The likelihood of a true result being returned. Must be in the interval `[0, 1]`.

Definition at line 3277 of file `random.h`.

5.379.4 Member Function Documentation

5.379.4.1 `result_type std::bernoulli_distribution::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3327 of file `random.h`.

5.379.4.2 `result_type std::bernoulli_distribution::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3320 of file `random.h`.

5.379.4.3 `template<typename _UniformRandomNumberGenerator> result_type std::bernoulli_distribution::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3335 of file `random.h`.

References `operator()()`, and `param()`.

Referenced by `operator()()`.

5.379.4.4 `double std::bernoulli_distribution::p () const [inline]`

Returns the `p` parameter of the distribution.

Definition at line 3298 of file `random.h`.

5.379.4.5 `void std::bernoulli_distribution::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3313 of file `random.h`.

5.379.4.6 `param_type std::bernoulli_distribution::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3305 of file `random.h`.

Referenced by `operator()()`, `std::operator==()`, and `std::operator>>()`.

5.379.4.7 `void std::bernoulli_distribution::reset () [inline]`

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3292 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

5.380 `std::bernoulli_distribution::param_type` Struct Reference

Public Types

- typedef [bernoulli_distribution](#) `distribution_type`

Public Member Functions

- `param_type` (double `__p`=0.5)
- double `p` () const

Friends

- `bool operator==(const param_type &__p1, const param_type &__p2)`

5.380.1 Detailed Description

Parameter type.

Definition at line 3246 of file `random.h`.

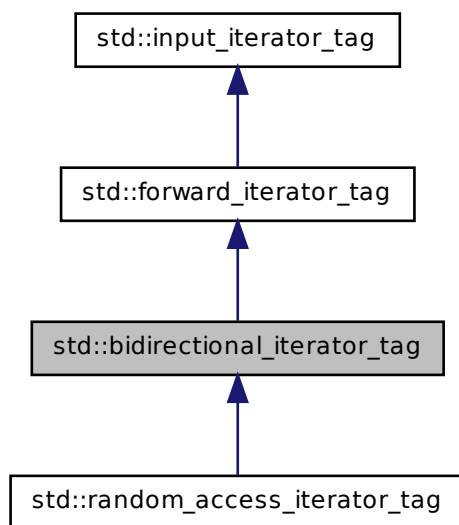
The documentation for this struct was generated from the following file:

- [random.h](#)

5.381 `std::bidirectional_iterator_tag` Struct Reference

Bidirectional iterators support a superset of forward iterator /// operations.

Inheritance diagram for `std::bidirectional_iterator_tag`:



5.381.1 Detailed Description

Bidirectional iterators support a superset of forward iterator `///` operations.

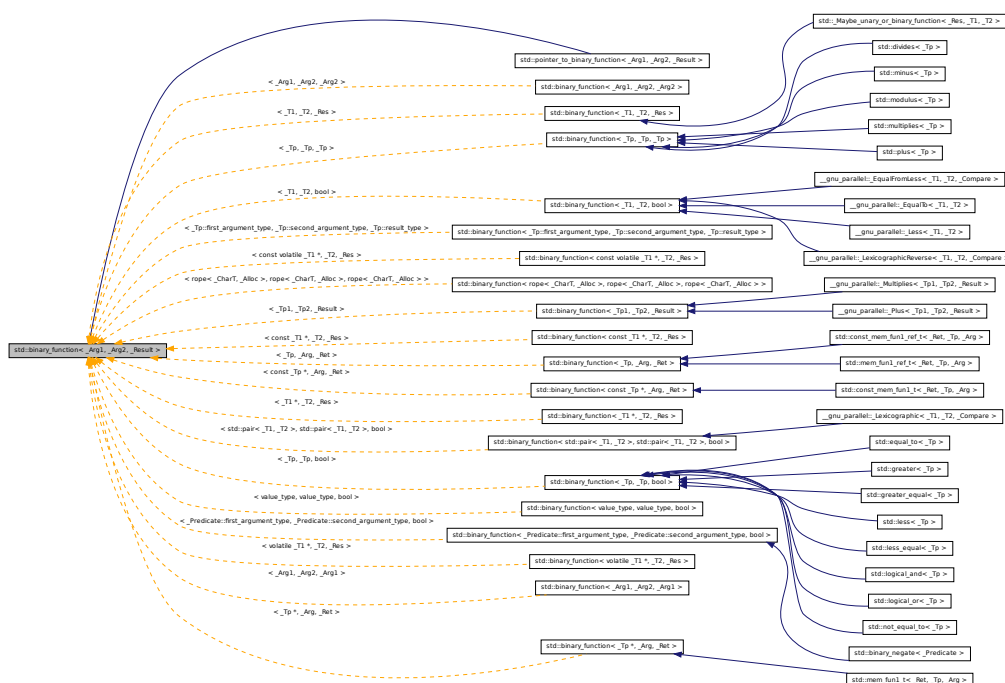
Definition at line 88 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- `stl_iterator_base_types.h`

5.382 `std::binary_function< _Arg1, _Arg2, _Result >` Struct Template Reference

Inheritance diagram for `std::binary_function<_Arg1, _Arg2, _Result>`:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type

- `typedef _Arg2 second_argument_type`

5.382.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> struct
std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 112 of file `stl_function.h`.

5.382.2 Member Typedef Documentation

5.382.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`
`>::first_argument_type`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.382.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`
`>::result_type`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.382.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`
`>::second_argument_type`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.383 std::binary_negate< _Predicate > Class Template Reference

One of the [negation functors](#).

Inheritance diagram for std::binary_negate< _Predicate >:



Public Types

- typedef _Predicate::first_argument_type [first_argument_type](#)
- typedef bool [result_type](#)
- typedef _Predicate::second_argument_type [second_argument_type](#)

Public Member Functions

- **binary_negate** (const _Predicate &__x)
- bool **operator()** (const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y) const

Protected Attributes

- _Predicate **_M_pred**

5.383.1 Detailed Description

```
template<typename _Predicate> class std::binary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 369 of file stl_function.h.

5.383.2 Member Typedef Documentation

5.383.2.1 typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::first_argument_type **[inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.383.2.2 `typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.383.2.3 `typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

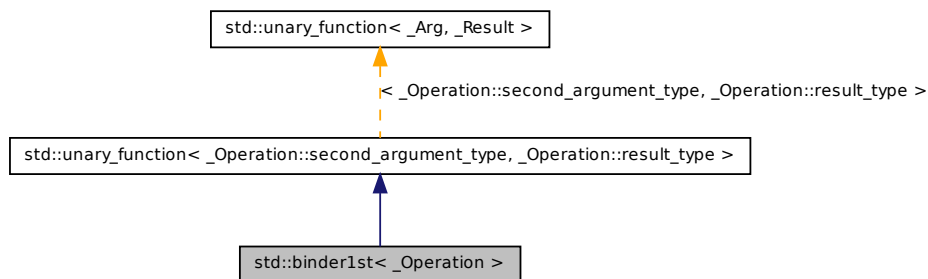
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.384 std::binder1st< _Operation > Class Template Reference

One of the [binder functors](#).

Inheritance diagram for std::binder1st< _Operation >:



Public Types

- typedef _Operation::second_argument_type [argument_type](#)
- typedef _Operation::result_type [result_type](#)

Public Member Functions

- **binder1st** (const _Operation &__x, const typename _Operation::first_argument_type &__y)
- _Operation::result_type **operator()** (typename _Operation::second_argument_type &__x) const
- _Operation::result_type **operator()** (const typename _Operation::second_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::first_argument_type **value**

5.384.1 Detailed Description

template<typename _Operation> class std::binder1st< _Operation >

One of the [binder functors](#).

Definition at line 98 of file binders.h.

5.384.2 Member Typedef Documentation

5.384.2.1 typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type, _Operation::result_type >::argument_type **[inherited]**

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.384.2.2 typedef _Operation::result_type std::unary_function< _Operation::second_argument_type, _Operation::result_type >::result_type **[inherited]**

`result_type` is the return type

Definition at line 105 of file stl_function.h.

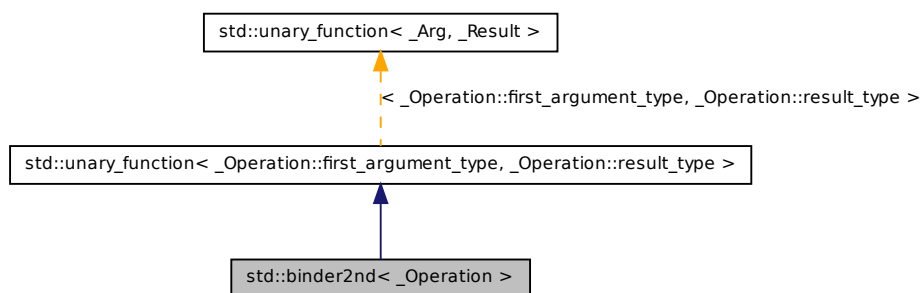
The documentation for this class was generated from the following file:

- [binders.h](#)

5.385 std::binder2nd< _Operation > Class Template Reference

One of the [binder functors](#).

Inheritance diagram for std::binder2nd< _Operation >:



Public Types

- typedef `_Operation::first_argument_type` [argument_type](#)
- typedef `_Operation::result_type` [result_type](#)

Public Member Functions

- **binder2nd** (`const _Operation &__x`, `const typename _Operation::second_argument_type &__y`)
- `_Operation::result_type` **operator()** (`typename _Operation::first_argument_type &__x`) `const`
- `_Operation::result_type` **operator()** (`const typename _Operation::first_argument_type &__x`) `const`

Protected Attributes

- `_Operation op`
- `_Operation::second_argument_type value`

5.385.1 Detailed Description

`template<typename _Operation> class std::binder2nd< _Operation >`

One of the [binder functors](#).

Definition at line 133 of file `binders.h`.

5.385.2 Member Typedef Documentation

5.385.2.1 `typedef _Operation::first_argument_type std::unary_function< _Operation::first_argument_type , _Operation::result_type >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.385.2.2 `typedef _Operation::result_type std::unary_function< _Operation::first_argument_type , _Operation::result_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

5.386 `std::binomial_distribution< _IntType >` Class Template Reference

A discrete binomial random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- **binomial_distribution** (`_IntType __t=_IntType(1), double __p=0.5`)
- **binomial_distribution** (const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng, const`
`param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- double **p** () const
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void [reset](#) ()
- `_IntType` **t** () const

Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
[std::basic_ostream](#)< `_CharT`, `_Traits` > & **operator<<** ([std::basic_ostream](#)< `_`
`CharT`, `_Traits` > &, const [std::binomial_distribution](#)< `_IntType1` > &)
- template<typename `_IntType1` >
bool **operator==** (const [std::binomial_distribution](#)< `_IntType1` > &__d1, const
[std::binomial_distribution](#)< `_IntType1` > &__d2)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
[std::basic_istream](#)< `_CharT`, `_Traits` > & **operator>>** ([std::basic_istream](#)< `_`
`CharT`, `_Traits` > &, [std::binomial_distribution](#)< `_IntType1` > &)

5.386.1 Detailed Description

```
template<typename _IntType = int> class std::binomial_distribution< _IntType
>
```

A discrete binomial random number distribution. The formula for the binomial probability density function is $p(i|t, p) = \binom{n}{i} p^i (1 - p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 3417 of file random.h.

5.386.2 Member Typedef Documentation

5.386.2.1 `template<typename _IntType = int> typedef _IntType
std::binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3424 of file random.h.

5.386.3 Member Function Documentation

5.386.3.1 `template<typename _IntType = int> result_type
std::binomial_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3527 of file random.h.

Referenced by std::binomial_distribution< _IntType >::operator().

5.386.3.2 `template<typename _IntType = int> result_type
std::binomial_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3520 of file random.h.

5.386.3.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::binomial_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3535 of file random.h.

References std::binomial_distribution< _IntType >::operator(), and std::binomial_distribution< _IntType >::param().

Referenced by std::binomial_distribution< _IntType >::operator().

5.386.3.4 `template<typename _IntType > template<typename
_UniformRandomNumberGenerator > binomial_distribution<
_IntType >::result_type std::binomial_distribution< _IntType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

A rejection algorithm when $t * p \geq 8$ and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1424 of file `random.tcc`.

References `std::abs()`, `std::log()`, and `std::binomial_distribution< _IntType >::max()`.

5.386.3.5 `template<typename _IntType = int> double
std::binomial_distribution< _IntType >::p () const [inline]`

Returns the distribution `p` parameter.

Definition at line 3498 of file `random.h`.

5.386.3.6 `template<typename _IntType = int> void std::binomial_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3513 of file `random.h`.

5.386.3.7 `template<typename _IntType = int> param_type
std::binomial_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 3505 of file `random.h`.

Referenced by `std::binomial_distribution< _IntType >::operator()()`.

5.386.3.8 `template<typename _IntType = int> void
std::binomial_distribution< _IntType >::reset() [inline]`

Resets the distribution state.

Definition at line 3484 of file random.h.

References std::normal_distribution< _RealType >::reset().

5.386.3.9 `template<typename _IntType = int> _IntType
std::binomial_distribution< _IntType >::t() const [inline]`

Returns the distribution t parameter.

Definition at line 3491 of file random.h.

5.386.4 Friends And Related Function Documentation

5.386.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream< _CharT ,
_Traits>& operator<< (std::basic_ostream< _CharT , _Traits > & ,
const std::binomial_distribution< _IntType1 > &) [friend]`

Inserts a binomial_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A binomial_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.386.4.2 `template<typename _IntType = int> template<typename _IntType1
> bool operator==(const std::binomial_distribution< _IntType1 >
& __d1, const std::binomial_distribution< _IntType1 > & __d2)
[friend]`

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3550 of file random.h.

5.386.4.3 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream<_CharT, _Traits > & ,
std::binomial_distribution<_IntType1 > &) [friend]`

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `binomial_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.387 `std::binomial_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `binomial_distribution<_IntType>` `distribution_type`

Public Member Functions

- `param_type` (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- `double p () const`
- `_IntType t () const`

Friends

- class `binomial_distribution<_IntType>`
- `bool operator==` (`const param_type &__p1`, `const param_type &__p2`)

5.387.1 Detailed Description

template<typename _IntType = int> struct std::binomial_distribution< _IntType >::param_type

Parameter type.

Definition at line 3426 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.388 std::bitset< _Nb > Class Template Reference

The bitset class represents a *fixed-size* sequence of bits.

Inheritance diagram for std::bitset< _Nb >:



Classes

- class [reference](#)

Public Member Functions

- [bitset](#) ()
- [bitset](#) (unsigned long long __val)
- template<class _CharT, class _Traits, class _Alloc >
[bitset](#) (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__s, size_t __-
position, size_t __n)
- template<class _CharT, class _Traits, class _Alloc >
bitset (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__s, size_t __-
position, size_t __n, _CharT __zero, _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc >
[bitset](#) (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__s, size_t __-
position=0)
- [bitset](#) (const char *__str)
- size_t [_Find_first](#) () const

- `size_t Find_next (size_t __prev) const`
- `template<class _CharT, class _Traits >`
`void _M_copy_from_ptr (const _CharT *, size_t, size_t, size_t, _CharT, _-`
`CharT)`
- `template<class _CharT, class _Traits, class _Alloc >`
`void _M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc`
`> &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)`
- `template<class _CharT, class _Traits, class _Alloc >`
`void _M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc`
`> &__s, size_t __pos, size_t __n)`
- `template<class _CharT, class _Traits, class _Alloc >`
`void _M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &, _-`
`CharT, _CharT) const`
- `template<class _CharT, class _Traits, class _Alloc >`
`void _M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &__s)`
`const`
- `bool all () const`
- `bool any () const`
- `size_t count () const`
- `bitset< _Nb > & flip ()`
- `bitset< _Nb > & flip (size_t __position)`
- `bool none () const`
- `bitset< _Nb > operator~ () const`
- `bitset< _Nb > & reset ()`
- `bitset< _Nb > & reset (size_t __position)`
- `bitset< _Nb > & set ()`
- `bitset< _Nb > & set (size_t __position, bool __val=true)`
- `size_t size () const`
- `bool test (size_t __position) const`
- `template<class _CharT, class _Traits >`
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string ()`
`const`
- `template<class _CharT >`
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`
`> > to_string () const`
- `template<class _CharT >`
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`
`> > to_string (_CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_string< _CharT, _Traits, _Alloc > to_string () const`
- `template<class _CharT, class _Traits >`
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string (_-`
`CharT __zero, _CharT __one=_CharT('1')) const`

- template<class _CharT, class _Traits, class _Alloc >
std::basic_string< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** (char __zero, char __one= '1') const
- std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** () const
- unsigned long long **to_ullong** () const
- unsigned long **to_ulong** () const
- bitset< _Nb > & **operator&=** (const bitset< _Nb > &__rhs)
- bitset< _Nb > & **operator|=** (const bitset< _Nb > &__rhs)
- bitset< _Nb > & **operator^=** (const bitset< _Nb > &__rhs)
- bitset< _Nb > & **operator<<=** (size_t __position)
- bitset< _Nb > & **operator>>=** (size_t __position)
- bitset< _Nb > & **_Unchecked_set** (size_t __pos)
- bitset< _Nb > & **_Unchecked_set** (size_t __pos, int __val)
- bitset< _Nb > & **_Unchecked_reset** (size_t __pos)
- bitset< _Nb > & **_Unchecked_flip** (size_t __pos)
- bool **_Unchecked_test** (size_t __pos) const
- reference **operator[]** (size_t __position)
- bool **operator[]** (size_t __position) const
- bool **operator==** (const bitset< _Nb > &__rhs) const
- bool **operator!=** (const bitset< _Nb > &__rhs) const
- bitset< _Nb > **operator<<** (size_t __position) const
- bitset< _Nb > **operator>>** (size_t __position) const

Private Types

- typedef unsigned long **_WordT**

Private Member Functions

- size_t **_M_are_all_aux** () const
- void **_M_do_and** (const _Base_bitset< _Nw > &__x)
- size_t **_M_do_count** () const
- size_t **_M_do_find_first** (size_t __not_found) const
- size_t **_M_do_find_next** (size_t __prev, size_t __not_found) const

- void **_M_do_flip** ()
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const [_Base_bitset](#)< _Nw > &__x)
- void **_M_do_reset** ()
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const [_Base_bitset](#)< _Nw > &__x)
- const _WordT * **_M_getdata** () const
- _WordT **_M_getword** (size_t __pos) const
- _WordT & **_M_getword** (size_t __pos)
- _WordT & **_M_hiword** ()
- _WordT **_M_hiword** () const
- bool **_M_is_any** () const
- bool **_M_is_equal** (const [_Base_bitset](#)< _Nw > &__x) const

Static Private Member Functions

- static _WordT **_S_maskbit** (size_t __pos)
- static size_t **_S_whichbit** (size_t __pos)
- static size_t **_S_whichbyte** (size_t __pos)
- static size_t **_S_whichword** (size_t __pos)

Private Attributes

- _WordT [_M_w](#) [_Nw]

Friends

- class **hash**
- class **reference**

5.388.1 Detailed Description

template<size_t _Nb> class std::bitset< _Nb >

The bitset class represents a *fixed-size* sequence of bits. (Note that bitset does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then $(Nb+(B-1))/B$ words will be used for storage. $B - NbB$ bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as *a simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index $Nb-1$ in the *most significant / left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long        a = 'a';
    bitset<10>  b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch33s02.html> for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the `bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

Definition at line 708 of file `bitset`.

5.388.2 Constructor & Destructor Documentation

5.388.2.1 `template<size_t _Nb> std::bitset<_Nb>::bitset () [inline]`

All bits set to zero.

Definition at line 802 of file `bitset`.

5.388.2.2 `template<size_t _Nb> std::bitset<_Nb>::bitset (unsigned long long __val) [inline]`

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 807 of file `bitset`.

5.388.2.3 `template<size_t _Nb> template<class _CharT , class _Traits , class _Alloc > std::bitset<_Nb>::bitset (const std::basic_string<_CharT, _Traits, _Alloc> & __s, size_t __position = 0) [inline, explicit]`

Use a subset of a string.

Parameters

s A string of *0* and *1* characters.

position Index of the first character in *s* to use; defaults to zero.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of *s*.

[*std::invalid_argument*](#) If a character appears in the string which is neither *0* nor *1*.

Definition at line 825 of file `bitset`.

5.388.2.4 `template<size_t _Nb> template<class _CharT , class _Traits , class _Alloc > std::bitset<_Nb>::bitset (const std::basic_string<_CharT, _Traits, _Alloc> & __s, size_t __position, size_t __n) [inline]`

Use a subset of a string.

Parameters

s A string of *0* and *1* characters.

position Index of the first character in *s* to use.

n The number of characters to copy.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of *s*.

[*std::invalid_argument*](#) If a character appears in the string which is neither *0* nor *1*.

Definition at line 847 of file `bitset`.

5.388.2.5 `template<size_t _Nb> std::bitset< _Nb >::bitset (const char * __str
) [inline, explicit]`

Construct from a string.

Parameters

str A string of *0* and *1* characters.

Exceptions

[*std::invalid_argument*](#) If a character appears in the string which is neither *0* nor *1*.

Definition at line 879 of file `bitset`.

5.388.3 Member Function Documentation

5.388.3.1 `template<size_t _Nb> bool std::bitset< _Nb >::all () const
 [inline]`

Tests whether all the bits are on.

Returns

True if all the bits are set.

Definition at line 1266 of file `bitset`.

5.388.3.2 `template<size_t _Nb> bool std::bitset< _Nb >::any () const
 [inline]`

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

Definition at line 1274 of file `bitset`.

5.388.3.3 `template<size_t _Nb> size_t std::bitset<_Nb >::count () const`
`[inline]`

Returns the number of bits which are set.

Definition at line 1226 of file `bitset`.

5.388.3.4 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::flip (`
`size_t __position) [inline]`

Toggles a given bit to its opposite value.

Parameters

position The index of the bit.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Definition at line 1066 of file `bitset`.

5.388.3.5 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::flip ()`
`[inline]`

Toggles every bit to its opposite value.

Definition at line 1053 of file `bitset`.

5.388.3.6 `template<size_t _Nb> bool std::bitset<_Nb >::none () const`
`[inline]`

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

Definition at line 1282 of file `bitset`.

5.388.3.7 `template<size_t _Nb> bool std::bitset< _Nb >::operator!= (const
bitset< _Nb > & __rhs) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1241 of file `bitset`.

5.388.3.8 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::operator&= (const bitset< _Nb > & __rhs) [inline]`

Operations on bitsets.

Parameters

rhs A same-sized bitset.

These should be self-explanatory.

Definition at line 900 of file `bitset`.

5.388.3.9 `template<size_t _Nb> bitset<_Nb> std::bitset< _Nb
>::operator<< (size_t __position) const [inline]`

Self-explanatory.

Definition at line 1288 of file `bitset`.

5.388.3.10 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::operator<<= (size_t __position) [inline]`

Operations on bitsets.

Parameters

position The number of places to shift.

These should be self-explanatory.

Definition at line 929 of file `bitset`.

5.388.3.11 `template<size_t _Nb> bool std::bitset< _Nb >::operator== (const
bitset< _Nb > & __rhs) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1237 of file `bitset`.

5.388.3.12 `template<size_t _Nb> bitset<_Nb> std::bitset< _Nb
>::operator>> (size_t __position) const [inline]`

Self-explanatory.

Definition at line 1292 of file `bitset`.

5.388.3.13 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::operator>>= (size_t __position) [inline]`

Operations on bitsets.

Parameters

position The number of places to shift.

These should be self-explanatory.

Definition at line 942 of file `bitset`.

5.388.3.14 `template<size_t _Nb> bool std::bitset< _Nb >::operator[] (size_t
__position) const [inline]`

Array-indexing support.

Parameters

position Index into the `bitset`.

Returns

A `bool` for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1098 of file `bitset`.

5.388.3.15 `template<size_t _Nb> reference std::bitset< _Nb >::operator[] (size_t __position) [inline]`

Array-indexing support.

Parameters

position Index into the bitset.

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1094 of file `bitset`.

5.388.3.16 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::operator^= (const bitset< _Nb > & __rhs) [inline]`

Operations on bitsets.

Parameters

rhs A same-sized bitset.

These should be self-explanatory.

Definition at line 914 of file `bitset`.

5.388.3.17 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::operator|= (const bitset< _Nb > & __rhs) [inline]`

Operations on bitsets.

Parameters

rhs A same-sized bitset.

These should be self-explanatory.

Definition at line 907 of file `bitset`.

5.388.3.18 `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb >::operator~
() const [inline]`

See the no-argument [flip\(\)](#).

Definition at line 1075 of file `bitset`.

5.388.3.19 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::reset (
) [inline]`

Sets every bit to false.

Definition at line 1028 of file `bitset`.

5.388.3.20 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::reset (
size_t __position) [inline]`

Sets a given bit to false.

Parameters

position The index of the bit.

Exceptions

[std::out_of_range](#) If *pos* is bigger the size of the set.

Same as writing `set (pos, false)`.

Definition at line 1042 of file `bitset`.

5.388.3.21 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::set ()
[inline]`

Sets every bit to true.

Definition at line 1003 of file `bitset`.

5.388.3.22 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::set (
size_t __position, bool __val = true) [inline]`

Sets a given bit to a particular value.

Parameters

position The index of the bit.
val Either true or false, defaults to true.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Definition at line 1017 of file `bitset`.

5.388.3.23 `template<size_t _Nb> size_t std::bitset< _Nb >::size () const`
`[inline]`

Returns the total number of bits.

Definition at line 1231 of file `bitset`.

5.388.3.24 `template<size_t _Nb> bool std::bitset< _Nb >::test (size_t`
`__position) const [inline]`

Tests the value of a bit.

Parameters

position The index of a bit.

Returns

The value at *pos*.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Definition at line 1252 of file `bitset`.

5.388.3.25 `template<size_t _Nb> template<class _CharT , class _Traits , class`
`_Alloc > std::basic_string<_CharT, _Traits, _Alloc> std::bitset<`
`_Nb >::to_string () const [inline]`

Returns a character interpretation of the bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1128 of file `bitset`.

5.388.3.26 `template<size_t _Nb> unsigned long std::bitset< _Nb >::to_ulong () const [inline]`

Returns a numerical interpretation of the `bitset`.

Returns

The integral equivalent of the bits.

Exceptions

[*std::overflow_error*](#) If there are too many bits to be represented in an `unsigned long`.

Definition at line 1109 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

5.389 `std::bitset< _Nb >::reference` Class Reference

Public Member Functions

- `reference` ([bitset](#) &__b, size_t __pos)
- [reference](#) & `flip` ()
- `operator bool` () const
- [reference](#) & `operator=` (const [reference](#) &__j)
- [reference](#) & `operator=` (bool __x)
- `bool operator~` () const

Friends

- class `bitset`

5.389.1 Detailed Description

`template<size_t _Nb> class std::bitset<_Nb>::reference`

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

Definition at line 739 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

5.390 `std::cauchy_distribution<_RealType>` Class Template Reference

A [cauchy_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- `cauchy_distribution` (`_RealType` __a=`_RealType`(0), `_RealType` __b=`_RealType`(1))
- `cauchy_distribution` (const [param_type](#) &__p)
- `_RealType a` () const
- `_RealType b` () const
- [result_type max](#) () const
- [result_type min](#) () const

- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng, const
param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

5.390.1 Detailed Description

`template<typename _RealType = double> class std::cauchy_distribution< _RealType >`

A [cauchy_distribution](#) random number distribution. The formula for the normal probability mass function is $p(x|a, b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2704 of file random.h.

5.390.2 Member Typedef Documentation

5.390.2.1 `template<typename _RealType = double> typedef _RealType
std::cauchy_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2711 of file random.h.

5.390.3 Member Function Documentation

5.390.3.1 `template<typename _RealType = double> result_type
std::cauchy_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2795 of file random.h.

5.390.3.2 `template<typename _RealType = double> result_type
std::cauchy_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2788 of file random.h.

5.390.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::cauchy_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2803 of file random.h.

References `std::cauchy_distribution< _RealType >::operator()()`, and `std::cauchy_distribution< _RealType >::param()`.

Referenced by `std::cauchy_distribution< _RealType >::operator()()`.

5.390.3.4 `template<typename _RealType = double> void
std::cauchy_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2781 of file random.h.

5.390.3.5 `template<typename _RealType = double> param_type
std::cauchy_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2773 of file random.h.

Referenced by `std::cauchy_distribution< _RealType >::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.390.3.6 `template<typename _RealType = double> void
std::cauchy_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2755 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.391 `std::cauchy_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [cauchy_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.391.1 Detailed Description

`template<typename _RealType = double> struct std::cauchy_distribution< _RealType >::param_type`

Parameter type.

Definition at line 2713 of file random.h.

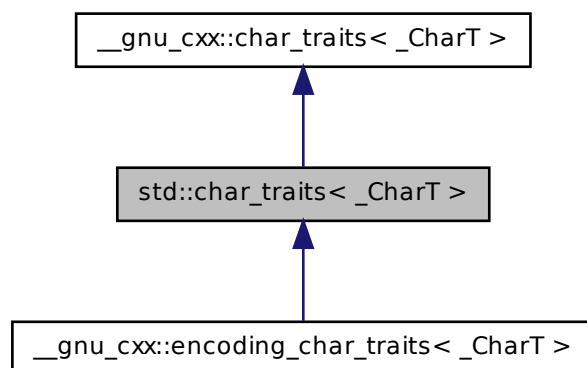
The documentation for this struct was generated from the following file:

- [random.h](#)

5.392 `std::char_traits< _CharT >` Struct Template Reference

Basis for explicit traits specializations.

Inheritance diagram for std::char_traits< _CharT >:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types< _CharT >::int_type` **int_type**
- typedef `_Char_types< _CharT >::off_type` **off_type**
- typedef `_Char_types< _CharT >::pos_type` **pos_type**
- typedef `_Char_types< _CharT >::state_type` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static int_type **eof** ()
- static bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static bool **eq_int_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)

- static std::size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.392.1 Detailed Description

template<class _CharT> struct std::char_traits< _CharT >

Basis for explicit traits specializations.

Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around [__gnu_cxx::char_traits](#), it is possible to achieve a more appropriate definition by specializing [__gnu_cxx::char_traits](#).

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod_char_traits.h](#).

Definition at line 231 of file char_traits.h.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.393 std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference

char_traits<__gnu_cxx::character> specialization.

Public Types

- typedef [__gnu_cxx::character](#)< V, I, S > **char_type**
- typedef char_type::int_type **int_type**
- typedef [streamoff](#) **off_type**
- typedef [fpos](#)< state_type > **pos_type**
- typedef char_type::state_type **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__i)
- static int_type **to_int_type** (const char_type &__c)

5.393.1 Detailed Description

template<typename V, typename I, typename S> struct std::char_traits< __gnu_cxx::character< V, I, S > >

char_traits<__gnu_cxx::character> specialization.

Definition at line 88 of file pod_char_traits.h.

The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

5.394 std::char_traits< char > Struct Template Reference

21.1.3.1 [char_traits](#) specializations

Public Types

- typedef char **char_type**
- typedef int **int_type**
- typedef [streamoff](#) **off_type**
- typedef [streampos](#) **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.394.1 Detailed Description

template<> struct std::char_traits< char >

21.1.3.1 [char_traits](#) specializations

Definition at line 237 of file char_traits.h.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.395 std::char_traits< wchar_t > Struct Template Reference

21.1.3.2 [char_traits](#) specializations

Public Types

- typedef wchar_t **char_type**
- typedef wint_t **int_type**
- typedef [streamoff](#) **off_type**
- typedef [wstreampos](#) **pos_type**
- typedef mbstate_t **state_type**

5.396 `std::chi_squared_distribution<_RealType>` Class Template Reference 2217

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.395.1 Detailed Description

`template<> struct std::char_traits< wchar_t >`

21.1.3.2 [char_traits](#) specializations

Definition at line 308 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.396 `std::chi_squared_distribution<_RealType>` Class Template Reference

A [chi_squared_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **chi_squared_distribution** (_RealType __n=_RealType(1))
- **chi_squared_distribution** (const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::chi_squared_distribution](#)< _RealType1 > &)
- template<typename _RealType1 >
bool [operator==](#) (const [std::chi_squared_distribution](#)< _RealType1 > &__d1, const [std::chi_squared_distribution](#)< _RealType1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::chi_squared_distribution](#)< _RealType1 > &)

5.396.1 Detailed Description

`template<typename _RealType = double> class std::chi_squared_distribution< _RealType >`

A [chi_squared_distribution](#) random number distribution. The formula for the normal probability mass function is $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Definition at line 2539 of file random.h.

5.396.2 Member Typedef Documentation

5.396.2.1 `template<typename _RealType = double> typedef _RealType std::chi_squared_distribution< _RealType >::result_type`

The type of the range of the distribution.

5.396 std::chi_squared_distribution< _RealType > Class Template Reference

Definition at line 2546 of file random.h.

5.396.3 Member Function Documentation

5.396.3.1 `template<typename _RealType = double> result_type
std::chi_squared_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2619 of file random.h.

5.396.3.2 `template<typename _RealType = double> result_type
std::chi_squared_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2612 of file random.h.

5.396.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::chi_squared_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2627 of file random.h.

5.396.3.4 `template<typename _RealType = double> param_type
std::chi_squared_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2597 of file random.h.

5.396.3.5 `template<typename _RealType = double> void
std::chi_squared_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 2605 of file random.h.

5.396.3.6 `template<typename _RealType = double> void
std::chi_squared_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2583 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

5.396.4 Friends And Related Function Documentation

5.396.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::chi_squared_distribution< _RealType1 > &) [friend]`

Inserts a `chi_squared_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

__os An output stream.

__x A `chi_squared_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.396.4.2 `template<typename _RealType = double> template<typename _
RealType1 > bool operator==(const std::chi_squared_distribution<
_RealType1 > & __d1, const std::chi_squared_distribution<
_RealType1 > & __d2) [friend]`

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2647 of file random.h.

5.396.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &, std::chi_squared_distribution<_RealType1> &) [friend]`

Extracts a `chi_squared_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `chi_squared_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.397 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [chi_squared_distribution<_RealType>](#) `distribution_type`

Public Member Functions

- `param_type` (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

Friends

- bool `operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.397.1 Detailed Description

```
template<typename _RealType = double> struct std::chi_squared_distribution<
_RealType >::param_type
```

Parameter type.

Definition at line 2548 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.398 std::chrono::duration< _Rep, _Period > Struct Template Reference

duration

Public Types

- typedef _Period **period**
- typedef _Rep **rep**

Public Member Functions

- template<typename _Rep2 , typename = typename enable_if<is_convertible<_Rep2, rep>::value && (treat_as_floating_point<rep>::value || !treat_as_floating_point<_Rep2>::value)>::type> **duration** (const _Rep2 &__rep)
- **duration** (const [duration](#) &)
- template<typename _Rep2 , typename _Period2 , typename = typename enable_if<treat_as_floating_point<rep>::value || (ratio_divide<_Period2, period>::type::den == 1 && !treat_as_floating_point<_Rep2>::value)>::type> **duration** (const [duration](#)< _Rep2, _Period2 > &__d)
- rep **count** () const
- template<typename _Rep2 = rep> [enable_if](#)<!treat_as_floating_point< _Rep2 >::value, [duration](#) & >::type **operator** %= (const rep &__rhs)
- template<typename _Rep2 = rep> [enable_if](#)<!treat_as_floating_point< _Rep2 >::value, [duration](#) & >::type **operator** %= (const [duration](#) &__d)
- [duration](#) & **operator** *= (const rep &__rhs)
- [duration](#) **operator** + () const

- `duration` & `operator++` ()
- `duration` `operator++` (int)
- `duration` & `operator+=` (const `duration` & __d)
- `duration` `operator-` () const
- `duration` & `operator--` ()
- `duration` `operator--` (int)
- `duration` & `operator-=` (const `duration` & __d)
- `duration` & `operator/=` (const rep & __rhs)
- `duration` & `operator=` (const `duration` &)
- `static_assert` (!__is_duration< _Rep >::value, "rep cannot be a `duration`")
- `static_assert` (_Period::num > 0, "period must be positive")
- `static_assert` (__is_ratio< _Period >::value, "period must be a specialization of `ratio`")

Static Public Member Functions

- static const `duration` `max` ()
- static const `duration` `min` ()
- static const `duration` `zero` ()

5.398.1 Detailed Description

`template<typename _Rep, typename _Period> struct std::chrono::duration< _Rep, _Period >`

`duration`

Definition at line 201 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

5.399 `std::chrono::duration_values< _Rep >` Struct Template Reference

`duration_values`

Static Public Member Functions

- static const `_Rep` **max** ()
- static const `_Rep` **min** ()
- static const `_Rep` **zero** ()

5.399.1 Detailed Description

`template<typename _Rep> struct std::chrono::duration_values< _Rep >`

[duration_values](#)

Definition at line 174 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

5.400 std::chrono::system_clock Struct Reference

[system_clock](#)

Public Types

- typedef [chrono::seconds](#) **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**
- typedef [chrono::time_point](#)< [system_clock](#), [duration](#) > **time_point**

Static Public Member Functions

- static [time_point](#) **from_time_t** (std::time_t __t)
- static [time_point](#) **now** () throw ()
- static std::time_t **to_time_t** (const [time_point](#) &__t)

Static Public Attributes

- static const bool **is_monotonic**

5.400.1 Detailed Description

[system_clock](#)

Definition at line 628 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

5.401 `std::chrono::time_point< _Clock, _Duration >` Struct Template Reference

[time_point](#)

Public Types

- typedef `_Clock` **clock**
- typedef `_Duration` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

Public Member Functions

- **time_point** (const `duration` &__dur)
- template<typename `_Duration2` >
 time_point (const [time_point](#)< `clock`, `_Duration2` > &__t)
- [time_point](#) & **operator+=** (const `duration` &__dur)
- [time_point](#) & **operator-=** (const `duration` &__dur)
- `duration` **time_since_epoch** () const

Static Public Member Functions

- static const [time_point](#) **max** ()
- static const [time_point](#) **min** ()

5.401.1 Detailed Description

```
template<typename _Clock, typename _Duration> struct std::chrono::time_
point< _Clock, _Duration >
```

[time_point](#)

Definition at line 492 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

5.402 `std::chrono::treat_as_floating_point< _Rep >` Struct Template Reference

[treat_as_floating_point](#)

Inherits `is_floating_point< _Rep >`.

5.402.1 Detailed Description

```
template<typename _Rep> struct std::chrono::treat_as_floating_point< _Rep >
```

[treat_as_floating_point](#)

Definition at line 168 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

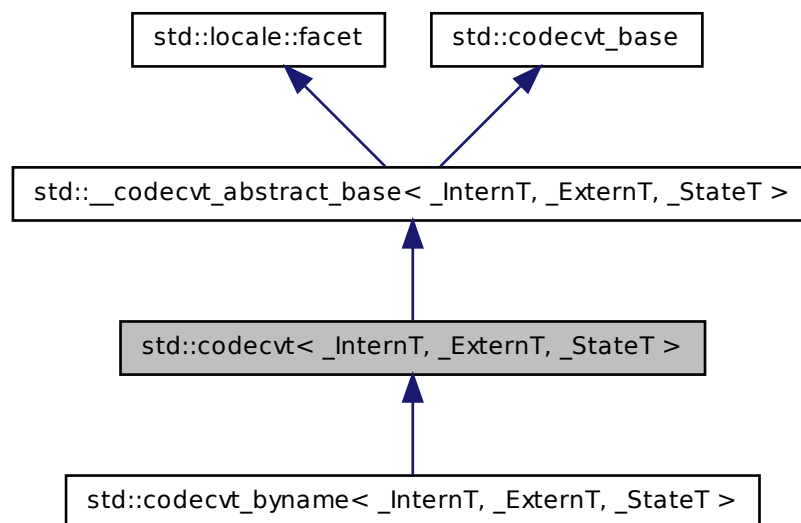
5.403 `std::codecvt< _InternT, _ExternT, _StateT >` Class Template Reference

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

5.403 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT, _ExternT, _StateT >`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`__c_locale __cloc, size_t __refs=0`)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (`state_type &__state, const extern_type *__from, const extern_type * __from_end, const extern_type *&__from_next, intern_type *__to, intern_type * __to_end, intern_type *&__to_next`) const

- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static **locale::id** id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class `locale::_Impl`

5.403.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt<_InternT, _ExternT, _StateT>`

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

Definition at line 275 of file `codecvt.h`.

5.403.2 Member Function Documentation

5.403.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt<_InternT, _ExternT, _StateT>::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const` `[protected, virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>`.

5.403.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const` `[inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.403.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

5.403 std::codecvt<_InternT, _ExternT, _StateT > Class Template Reference 2231

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

```
5.403.2.4 template<typename _InternT, typename _ExternT, typename
        _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
        _StateT >::unshift ( state_type & __state, extern_type * __to,
        extern_type * __to_end, extern_type *& __to_next ) const
        [inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.404 `std::codecvt< _InternT, _ExternT, encoding_state >` Class Template Reference

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Inheritance diagram for `std::codecvt< _InternT, _ExternT, encoding_state >`:



Public Types

- typedef `state_type::descriptor_type` **descriptor_type**

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (state_type &__enc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static `locale::id` **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()

- virtual result `do_out` (`state_type` &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result `do_out` (`state_type` &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result `do_unshift` (`state_type` &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result `do_unshift` (`state_type` &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale` &__cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` &__cloc, const char *__s, `__c_locale` __old=0)
- static void `_S_destroy_c_locale` (`__c_locale` &__cloc)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char * `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale` __cloc, const char *__s)

Friends

- class `locale::_Impl`

5.404.1 Detailed Description

`template<typename _InternT, typename _ExternT> class std::codecvt< _InternT, _ExternT, encoding_state >`

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Definition at line 227 of file `codecvt_specializations.h`.

5.404.2 Member Function Documentation

5.404.2.1 virtual result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const [protected, pure virtual, inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

5.404.2.2 result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next) const [inline, inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.

from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 195 of file codecvt.h.

5.404.2.3 `result std::__codecvt_abstract_base< _InternT, _ExternT,
 encoding_state >::out (state_type & __state, const intern_type
 * __from, const intern_type * __from_end, const intern_type
 *& __from_next, extern_type * __to, extern_type * __to_end,
 extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of *intern_type* to output string of *extern_type*. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. *from_next* and *to_next* are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, *from_next* and *to_next* are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

5.404.2.4 `result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 154 of file codecvt.h.

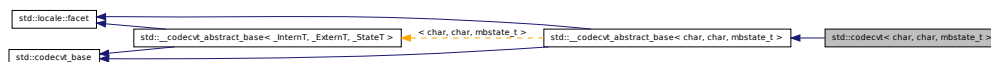
The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

5.405 std::codecvt< char, char, mbstate_t > Class Template Reference

class [codecvt<char, char, mbstate_t>](#) specialization.

Inheritance diagram for std::codecvt< char, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type * __from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type * __from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class `locale::_Impl`

5.405.1 Detailed Description

`template<> class std::codecvt< char, char, mbstate_t >`

class `codecvt<char, char, mbstate_t>` specialization.

Definition at line 337 of file `codecvt.h`.

5.405.2 Member Function Documentation

5.405.2.1 virtual result `std::__codecvt_abstract_base< char , char , mbstate_t >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const` [`protected`, `pure virtual`, `inherited`]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

out for more information.

5.405.2.2 result `std::__codecvt_abstract_base< char , char , mbstate_t >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const` [`inline`, `inherited`]

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted

character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

5.405.2.3 `result std::__codecvt_abstract_base< char , char , mbstate_t >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

5.405.2.4 `result std::__codecvt_abstract_base< char , char , mbstate_t
 >::unshift (state_type & __state, extern_type * __to, extern_type
 * __to_end, extern_type *& __to_next) const [inline,
 inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

5.406 std::codecvt< wchar_t, char, mbstate_t > Class Template Reference 2243

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 154 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.406 std::codecvt< wchar_t, char, mbstate_t > Class Template Reference

class [codecvt<wchar_t, char, mbstate_t>](#) specialization.

Inheritance diagram for std::codecvt< wchar_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef wchar_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- codecvt** (size_t __refs=0)
- codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()

- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class locale::_Impl

5.406.1 Detailed Description

template<> class std::codecvt< wchar_t, char, mbstate_t >

class [codecvt<wchar_t, char, mbstate_t>](#) specialization.

Definition at line 395 of file codecvt.h.

5.406.2 Member Function Documentation

5.406.2.1 virtual result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected, pure virtual, inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

```

5.406.2.2 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t
>::in ( state_type & __state, const extern_type * __from, const
extern_type * __from_end, const extern_type *& __from_next,
intern_type * __to, intern_type * __to_end, intern_type *&
__to_next ) const [inline, inherited]

```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

5.406.2.3 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

5.406.2.4 `result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

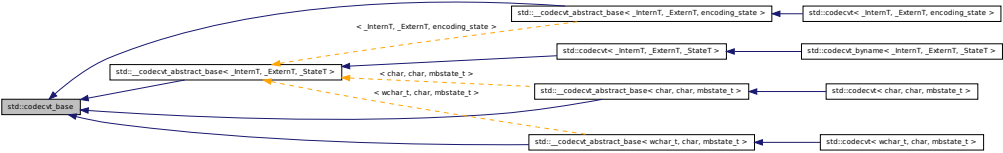
- [codecvt.h](#)

5.407 std::codecvt_base Class Reference

Empty base class for `codecvt` facet [22.2.1.5].

5.408 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference2249

Inheritance diagram for std::codecvt_base:



Public Types

- enum result { ok, partial, error, noconv }

5.407.1 Detailed Description

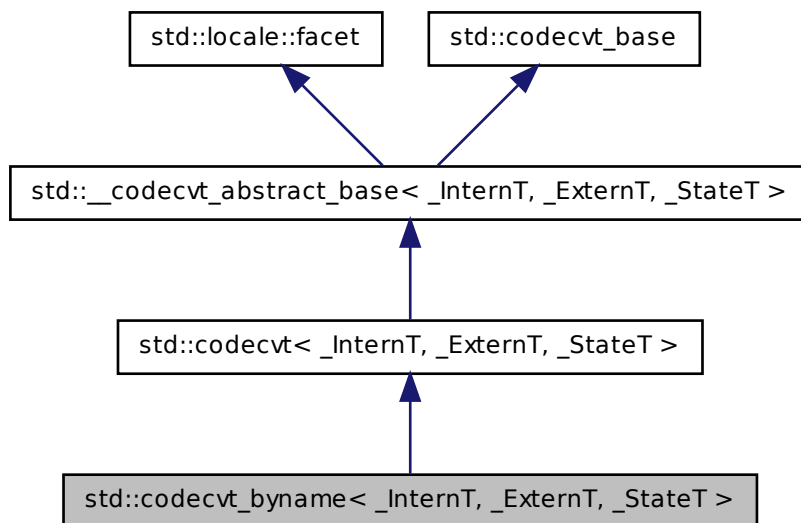
Empty base class for codecvt facet [22.2.1.5].
Definition at line 45 of file codecvt.h.
The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.408 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference

class [codecvt_byname](#) [22.2.1.6].

Inheritance diagram for `std::codecvt_byname< _InternT, _ExternT, _StateT >`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- **codecvt_byname** (const char *__s, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const

- `int max_length ()` `const throw ()`
- `result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const`
- `result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const`

Static Public Attributes

- `static locale::id id`

Protected Member Functions

- `virtual bool do_always_noconv ()` `const throw ()`
- `virtual int do_encoding ()` `const throw ()`
- `virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const`
- `virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const`
- `virtual int do_max_length ()` `const throw ()`
- `virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const`
- `virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const`

Static Protected Member Functions

- `static __c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- `static void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- `static void _S_destroy_c_locale (__c_locale &__cloc)`
- `static __c_locale _S_get_c_locale ()`
- `static _GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- `static __c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Protected Attributes

- `__c_locale _M_c_locale_codecvt`

Friends

- class `locale::_Impl`

5.408.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt_byname< _InternT, _ExternT, _StateT >`

class `codecvt_byname` [22.2.1.6].

Definition at line 455 of file `codecvt.h`.

5.408.2 Member Function Documentation

5.408.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected, virtual, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`.

5.408.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

5.408 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference 2253

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.408.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

```
5.408.2.4 template<typename _InternT, typename _ExternT, typename
    _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
    _StateT >::unshift ( state_type & __state, extern_type * __to,
    extern_type * __to_end, extern_type *& __to_next ) const
    [inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

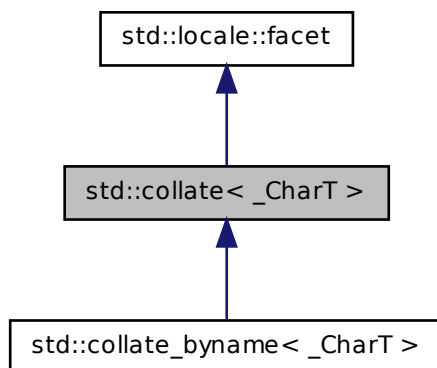
The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.409 `std::collate<_CharT>` Class Template Reference

Facet for localized string comparison.

Inheritance diagram for `std::collate<_CharT>`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef [basic_string<_CharT>](#) [string_type](#)

Public Member Functions

- [collate](#) (size_t __refs=0)
- [collate](#) (__c_locale __cloc, size_t __refs=0)
- template<>
int **_M_compare** (const char *, const char *) const throw()
- template<>
int **_M_compare** (const wchar_t *, const wchar_t *) const throw()
- int **_M_compare** (const _CharT *, const _CharT *) const throw ()
- template<>
size_t **_M_transform** (char *, const char *, size_t) const throw()
- template<>
size_t **_M_transform** (wchar_t *, const wchar_t *, size_t) const throw()
- size_t **_M_transform** (_CharT *, const _CharT *, size_t) const throw ()
- int [compare](#) (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const

- long [hash](#) (const _CharT *__lo, const _CharT *__hi) const
- [string_type transform](#) (const _CharT *__lo, const _CharT *__hi) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~collate](#) ()
- virtual int [do_compare](#) (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- virtual long [do_hash](#) (const _CharT *__lo, const _CharT *__hi) const
- virtual [string_type do_transform](#) (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale [_M_c_locale_collate](#)

Friends

- class [locale::_Impl](#)

5.409.1 Detailed Description

template<typename _CharT> class std::collate<_CharT>

Facet for localized string comparison. This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 610 of file locale_classes.h.

5.409.2 Member Typedef Documentation

5.409.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::collate_byname<_CharT>](#).

Definition at line 616 of file locale_classes.h.

5.409.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::collate_byname<_CharT>](#).

Definition at line 617 of file locale_classes.h.

5.409.3 Constructor & Destructor Documentation

5.409.3.1 `template<typename _CharT> std::collate<_CharT>::collate (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 637 of file locale_classes.h.

5.409.3.2 `template<typename _CharT> std::collate<_CharT>::collate (__c_locale __cloc, size_t __refs = 0) [inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.
refs Passed to the base facet class.

Definition at line 651 of file locale_classes.h.

5.409.3.3 `template<typename _CharT> virtual std::collate<_CharT>::~collate() [inline, protected, virtual]`

Destructor.

Definition at line 714 of file locale_classes.h.

5.409.4 Member Function Documentation

5.409.4.1 `template<typename _CharT> int std::collate<_CharT>::compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [inline]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do_compare\(\)](#).

Parameters

lo1 Start of string 1.
hi1 End of string 1.
lo2 Start of string 2.
hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 668 of file locale_classes.h.

5.409.4.2 `template<typename _CharT> int std::collate<_CharT>::do_compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [protected, virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

[compare\(\)](#).

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 134 of file locale_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

5.409.4.3 `template<typename _CharT> long std::collate<_CharT>
>::do_hash (const _CharT * __lo, const _CharT * __hi) const
[protected, virtual]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

lo Start of string.

hi End of string.

Returns

Hash value.

Definition at line 229 of file locale_classes.tcc.

5.409.4.4 `template<typename _CharT> collate<_CharT>::string_type
std::collate<_CharT>::do_transform (const _CharT * __lo, const
_CharT * __hi) const [protected, virtual]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

lo1 Start of string 1.
hi1 End of string 1.
lo2 Start of string 2.
hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 173 of file locale_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, `std::basic_string<_CharT, _Traits, _Alloc>::length()`, and `std::basic_string<_CharT, _Traits, _Alloc>::push_back()`.

5.409.4.5 `template<typename _CharT> long std::collate<_CharT>::hash (const _CharT * __lo, const _CharT * __hi) const [inline]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

Parameters

lo Start of string.
hi End of string.

Returns

Hash value.

Definition at line 701 of file locale_classes.h.

5.409.4.6 `template<typename _CharT> string_type std::collate<_CharT>::transform (const _CharT * __lo, const _CharT * __hi) const [inline]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

Parameters

lo Start of string.

hi End of string.

Returns

Transformed string_type.

Definition at line 687 of file locale_classes.h.

Referenced by std::regex_traits< _Ch_type >::transform().

5.409.5 Member Data Documentation

5.409.5.1 `template<typename _CharT> locale::id std::collate< _CharT >::id` `[static]`

Numpunct facet id.

Definition at line 627 of file locale_classes.h.

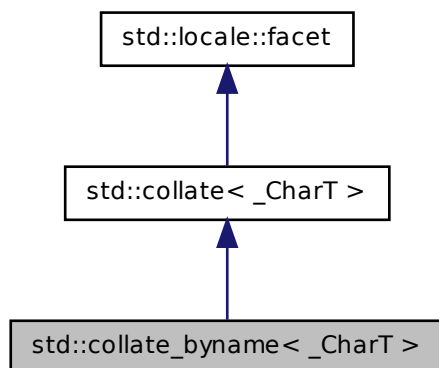
The documentation for this class was generated from the following files:

- [locale_classes.h](#)
- [locale_classes.tcc](#)

5.410 `std::collate_byname< _CharT >` Class Template Reference

class [collate_byname](#) [22.2.4.2].

Inheritance diagram for std::collate_byname< _CharT >:



Public Types

- typedef `_CharT` [char_type](#)
- typedef [basic_string](#)< `_CharT` > [string_type](#)

Public Member Functions

- **collate_byname** (const char *__s, size_t __refs=0)
- **int _M_compare** (const _CharT *, const _CharT *) const throw ()
- template<>
 int _M_compare (const char *, const char *) const throw()
- template<>
 int _M_compare (const wchar_t *, const wchar_t *) const throw()
- **size_t _M_transform** (_CharT *, const _CharT *, size_t) const throw ()
- template<>
 size_t _M_transform (char *, const char *, size_t) const throw()
- template<>
 size_t _M_transform (wchar_t *, const wchar_t *, size_t) const throw()
- **int compare** (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- **long hash** (const _CharT *__lo, const _CharT *__hi) const
- **string_type transform** (const _CharT *__lo, const _CharT *__hi) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual int [do_compare](#) (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- virtual long [do_hash](#) (const _CharT *__lo, const _CharT *__hi) const
- virtual [string_type do_transform](#) (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale [_M_c_locale_collate](#)

Friends

- class [locale::_Impl](#)

5.410.1 Detailed Description

`template<typename _CharT> class std::collate_byname< _CharT >`

class [collate_byname](#) [22.2.4.2].

Definition at line 786 of file `locale_classes.h`.

5.410.2 Member Typedef Documentation

5.410.2.1 `template<typename _CharT > typedef _CharT std::collate_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::collate< _CharT >](#).

Definition at line 791 of file locale_classes.h.

5.410.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::collate< _CharT >](#).

Definition at line 792 of file locale_classes.h.

5.410.3 Member Function Documentation

5.410.3.1 `template<typename _CharT> int std::collate< _CharT >::compare (const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [inline, inherited]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do_compare\(\)](#).

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 668 of file locale_classes.h.

5.410.3.2 `template<typename _CharT> int std::collate< _CharT
>::do_compare (const _CharT * __lo1, const _CharT *
__hi1, const _CharT * __lo2, const _CharT * __hi2) const
[protected, virtual, inherited]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

[compare\(\)](#).

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 134 of file locale_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::length()`.

5.410.3.3 `template<typename _CharT> long std::collate< _CharT
>::do_hash (const _CharT * __lo, const _CharT * __hi) const
[protected, virtual, inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

lo Start of string.

hi End of string.

Returns

Hash value.

Definition at line 229 of file locale_classes.tcc.

5.410.3.4 `template<typename _CharT> collate< _CharT >::string_type
std::collate< _CharT >::do_transform (const _CharT * __lo, const
_CharT * __hi) const [protected, virtual, inherited]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 173 of file locale_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, and `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`.

5.410.3.5 `template<typename _CharT> long std::collate< _CharT >::hash (
const _CharT * __lo, const _CharT * __hi) const [inline,
inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do_hash\(\)](#).

Parameters

lo Start of string.

hi End of string.

Returns

Hash value.

Definition at line 701 of file locale_classes.h.

5.410.3.6 `template<typename _CharT> string_type std::collate< _CharT
>::transform (const _CharT * __lo, const _CharT * __hi) const
[inline, inherited]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do_transform\(\)](#).

Parameters

lo Start of string.

hi End of string.

Returns

Transformed `string_type`.

Definition at line 687 of file `locale_classes.h`.

Referenced by `std::regex_traits< _Ch_type >::transform()`.

5.410.4 Member Data Documentation

5.410.4.1 `template<typename _CharT> locale::id std::collate< _CharT >::id
[static, inherited]`

Numpunct facet id.

Definition at line 627 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

5.411 std::complex< _Tp > Struct Template Reference

Public Types

- `typedef _Tp value_type`

Public Member Functions

- `complex` (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())
- `template<typename _Up >`
`complex` (const `complex`< _Up > &__z)
- `const complex & __rep () const`
- `_Tp imag () const`
- `void imag (_Tp __val)`
- `complex< _Tp > & operator*= (const _Tp &)`
- `template<typename _Up >`
`complex< _Tp > & operator*= (const complex< _Up > &)`
- `complex< _Tp > & operator+= (const _Tp &__t)`
- `template<typename _Up >`
`complex< _Tp > & operator+= (const complex< _Up > &)`
- `complex< _Tp > & operator-= (const _Tp &__t)`
- `template<typename _Up >`
`complex< _Tp > & operator-= (const complex< _Up > &)`
- `template<typename _Up >`
`complex< _Tp > & operator/= (const complex< _Up > &)`
- `complex< _Tp > & operator/= (const _Tp &)`
- `template<typename _Up >`
`complex< _Tp > & operator= (const complex< _Up > &)`
- `complex< _Tp > & operator= (const _Tp &)`
- `void real (_Tp __val)`
- `_Tp real () const`

5.411.1 Detailed Description

`template<typename _Tp> struct std::complex< _Tp >`

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

Parameters

Tp Type of real and imaginary values.

Definition at line 122 of file `complex`.

5.411.2 Member Typedef Documentation

5.411.2.1 `template<typename _Tp> typedef _Tp std::complex< _Tp >::value_type`

Value typedef.

Definition at line 125 of file `complex`.

5.411.3 Constructor & Destructor Documentation

5.411.3.1 `template<typename _Tp> std::complex< _Tp >::complex (const _Tp & __r = _Tp (), const _Tp & __i = _Tp ()) [inline]`

Default constructor. First parameter is x, second parameter is y. /// Unspecified parameters default to 0.

Definition at line 129 of file `complex`.

5.411.3.2 `template<typename _Tp> template<typename _Up > std::complex< _Tp >::complex (const complex< _Up > & __z) [inline]`

Copy constructor.

Definition at line 136 of file `complex`.

5.411.4 Member Function Documentation

5.411.4.1 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator+=(const _Tp & __t) [inline]`

Add *t* to this complex number.

Definition at line 179 of file `complex`.

5.411.4.2 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator-= (const _Tp & __t) [inline]`

Subtract *t* from this complex number.

Definition at line 188 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

5.412 std::condition_variable Class Reference

[condition_variable](#)

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **condition_variable** (const [condition_variable](#) &)
- native_handle_type **native_handle** ()
- void **notify_all** ()
- void **notify_one** ()
- [condition_variable](#) & **operator=** (const [condition_variable](#) &)
- template<typename _Predicate >
void **wait** ([unique_lock](#)< [mutex](#) > &__lock, _Predicate __p)
- void **wait** ([unique_lock](#)< [mutex](#) > &__lock)
- template<typename _Rep, typename _Period, typename _Predicate >
bool **wait_for** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Rep, typename _Period >
[cv_status](#) **wait_for** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Duration >
[cv_status](#) **wait_until** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::time_point](#)< __clock_t, _Duration > &__atime)
- template<typename _Clock, typename _Duration >
[cv_status](#) **wait_until** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- template<typename _Clock, typename _Duration, typename _Predicate >
bool **wait_until** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime, _Predicate __p)

5.412.1 Detailed Description

[condition_variable](#)

Definition at line 57 of file condition_variable.

The documentation for this class was generated from the following file:

- [condition_variable](#)

5.413 std::condition_variable_any Class Reference

[condition_variable_any](#)

Public Types

- typedef condition_variable::native_handle_type **native_handle_type**

Public Member Functions

- **condition_variable_any** (const [condition_variable_any](#) &)
- native_handle_type **native_handle** ()
- void **notify_all** ()
- void **notify_one** ()
- [condition_variable_any](#) & **operator=** (const [condition_variable_any](#) &)
- template<typename _Lock , typename _Predicate >
void **wait** (_Lock &__lock, _Predicate __p)
- template<typename _Lock >
void **wait** (_Lock &__lock)
- template<typename _Lock , typename _Rep , typename _Period , typename _Predicate >
bool **wait_for** (_Lock &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Lock , typename _Rep , typename _Period >
[cv_status](#) **wait_for** (_Lock &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Lock , typename _Clock , typename _Duration >
[cv_status](#) **wait_until** (_Lock &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- template<typename _Lock , typename _Clock , typename _Duration , typename _Predicate >
bool **wait_until** (_Lock &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime, _Predicate __p)

5.413.1 Detailed Description

[condition_variable_any](#)

Definition at line 166 of file condition_variable.

The documentation for this class was generated from the following file:

- [condition_variable](#)

5.414 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference

`conditional`

Public Types

- `typedef _Iftrue type`

5.414.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse> struct  
std::conditional< _Cond, _Iftrue, _Iffalse >
```

`conditional`

Definition at line 397 of file `type_traits`.

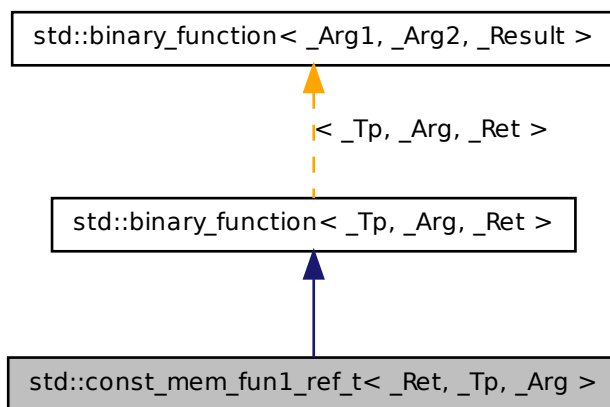
The documentation for this struct was generated from the following file:

- [type_traits](#)

5.415 `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `const_mem_fun1_ref_t` (`_Ret(_Tp::*__pf)(_Arg) const`)
- `_Ret operator()` (`const _Tp &__r, _Arg __x`) const

5.415.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`

One of the [adaptors for member /// pointers](#).

Definition at line 650 of file `stl_function.h`.

5.416 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

5.415.2 Member Typedef Documentation

5.415.2.1 `typedef _Tp std::binary_function< _Tp , _Arg , _Ret
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.415.2.2 `typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.415.2.3 `typedef _Arg std::binary_function< _Tp , _Arg , _Ret
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

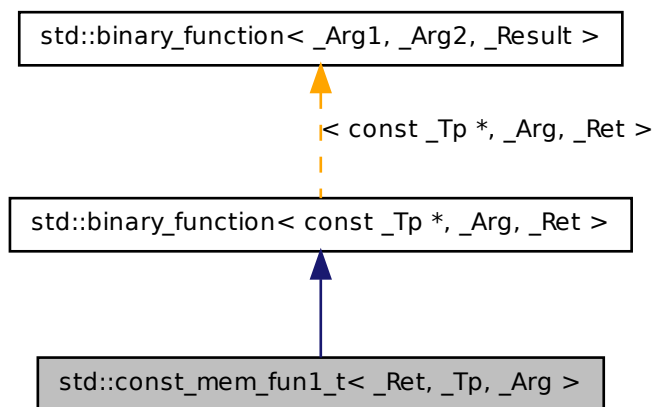
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.416 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`:



Public Types

- typedef const _Tp * [first_argument_type](#)
- typedef _Ret [result_type](#)
- typedef _Arg [second_argument_type](#)

Public Member Functions

- **const_mem_fun1_t** (_Ret(_Tp::*__pf)(_Arg) const)
- _Ret **operator()** (const _Tp *__p, _Arg __x) const

5.416.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_t<_Ret, _Tp, _Arg>`

One of the [adaptors for member /// pointers](#).

Definition at line 614 of file `stl_function.h`.

5.416.2 Member Typedef Documentation

5.416.2.1 `typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.416.2.2 `typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.416.2.3 `typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

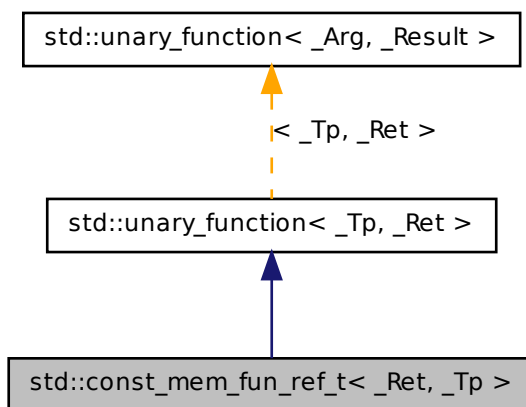
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.417 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_ref_t< _Ret, _Tp >`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `const_mem_fun_ref_t` (`_Ret`(`_Tp::*__pf`)() const)
- `_Ret operator()` (const `_Tp` &`__r`) const

5.417.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::const_mem_fun_ref_t< _Ret, _Tp >`

One of the [adaptors for member /// pointers](#).

Definition at line 578 of file `stl_function.h`.

5.417.2 Member Typedef Documentation

5.417.2.1 `typedef _Tp std::unary_function<_Tp, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.417.2.2 `typedef _Ret std::unary_function<_Tp, _Ret>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

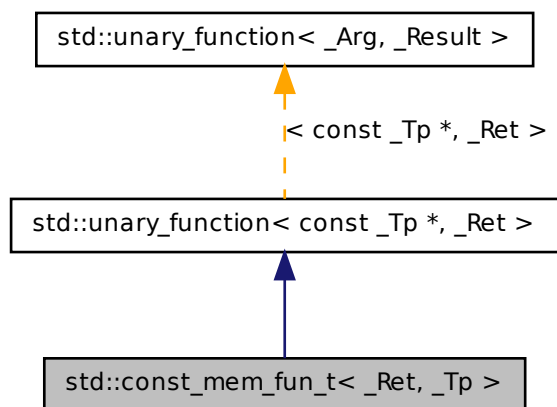
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.418 `std::const_mem_fun_t<_Ret, _Tp>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_t< _Ret, _Tp >`:



Public Types

- typedef `const _Tp *` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `const_mem_fun_t` (`_Ret(_Tp::*__pf)()` `const`)
- `_Ret operator()` (`const _Tp *__p`) `const`

5.418.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::const_mem_fun_t< _Ret,
_Tp >
```

One of the [adaptors for member /// pointers](#).

Definition at line 542 of file `stl_function.h`.

5.418.2 Member Typedef Documentation

5.418.2.1 `typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.418.2.2 `typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

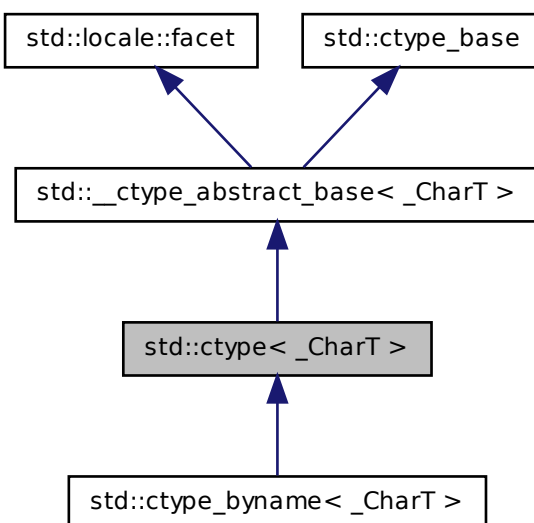
- [stl_function.h](#)

5.419 `std::ctype<_CharT>` Class Template Reference

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets. It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

Inheritance diagram for `std::ctype<_CharT>`:



Public Types

- typedef const int * **__to_type**
- typedef `_CharT` **char_type**
- typedef `__ctype_abstract_base<_CharT>::mask` **mask**

Public Member Functions

- **ctype** (size_t __refs=0)
- bool **is** (mask __m, **char_type** __c) const
- const **char_type** * **is** (const **char_type** *__lo, const **char_type** *__hi, mask *__-vec) const
- const **char_type** * **narrow** (const **char_type** *__lo, const **char_type** *__hi, **char** __default, **char** *__to) const
- **char** **narrow** (**char_type** __c, **char** __default) const
- const **char_type** * **scan_is** (mask __m, const **char_type** *__lo, const **char_type** *__hi) const

- const [char_type](#) * [scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __dfault, char * __dest) const
- virtual char [do_narrow](#) ([char_type](#), char __dfault) const
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#) __c) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#) __c) const

- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const
- virtual [char_type do_widen](#) (char __c) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.419.1 Detailed Description

template<typename _CharT> class std::ctype< _CharT >

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations. This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in [__ctype_abstract_base](#), to allow for implementation flexibility. See [ctype<wchar_t>](#) for an example. The functions are documented in [__ctype_abstract_base](#).

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 604 of file locale_facets.h.

5.419.2 Member Typedef Documentation

5.419.2.1 **template<typename _CharT> typedef _CharT std::ctype< _CharT >::char_type**

Typedef for the template parameter.

Reimplemented from [std::__ctype_abstract_base< _CharT >](#).

Definition at line 608 of file locale_facets.h.

5.419.3 Member Function Documentation

5.419.3.1 `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const [protected, virtual]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.2 `template<typename _CharT> virtual bool std::ctype< _CharT >::do_is (mask __m, char_type __c) const [protected, virtual]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

c The char_type to find the mask of.

m The mask to compare against.

Returns

(M & m) != 0.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.419.3.3 `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow (char_type, char __dfault) const [protected, virtual]`

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead.

[do_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char_type to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.419.3.4 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, virtual]`

Narrow char_type array to char.

This virtual function converts each char_type in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

[do_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

dfault Char to use if conversion fails.

to Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.7 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower (char_type * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.8 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower (char_type) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

- c* The `char_type` to convert.

Returns

The lowercase char_type if convertible, else *c*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.9 `template<typename _CharT> virtual const char_type* std::ctype<
_CharT >::do_toupper(char_type * __lo, const char_type * __hi)
const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.10 `template<typename _CharT> virtual char_type std::ctype<
_CharT >::do_toupper(char_type) const [protected,
virtual]`

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The char_type to convert.

Returns

The uppercase char_type if convertible, else *c*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.419.3.11 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen (char) const [protected, virtual]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type

Implements [std::__ctype_abstract_base<_CharT>](#).

5.419.3.12 `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, virtual]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.419.3.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is(mask __m, char_type __c) const [inline, inherited]`

Test char_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

c The char_type to compare the mask of.

m The mask to compare against.

Returns

(*M* & *m*) != 0.

Definition at line 161 of file locale_facets.h.

Referenced by `std::regex_traits< _Ch_type >::isctype()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.419.3.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is(const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.419.3.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow (char_type __c, char __dfault) const
[inline, inherited]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

Returns

The converted `char`.

Definition at line 323 of file `locale_facets.h`.

Referenced by `std::time_put<_CharT, _OutIter>::put()`.

5.419.3.16 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::narrow (const char_type
* __lo, const char_type * __hi, char __dfault, char * __to) const
[inline, inherited]`

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

```
5.419.3.17 template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char_type>::do_scan_is\(\)](#).

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

```
5.419.3.18 template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::scan_not ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char_type>::do_scan_not\(\)](#).

Parameters

m The mask to compare against.

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.419.3.19 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::tolower (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file `locale_facets.h`.

5.419.3.20 `template<typename _CharT> char_type std::__ctype_abstract_-
base<_CharT>::tolower (char_type __c) const [inline,
inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper(c)`.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

5.419.3.21 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::toupper (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.419.3.22 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.419.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen (char __c) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type.

Definition at line 285 of file locale_facets.h.

Referenced by std::money_get<_CharT, _InIter>::do_get(), std::time_put<_CharT, _OutIter>::do_put(), std::money_put<_CharT, _OutIter>::do_put(), std::regex_traits<_Ch_type>::isctype(), and std::operator<<().

5.419.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file locale_facets.h.

5.419.4 Member Data Documentation

5.419.4.1 `template<typename _CharT> locale::id std::ctype<_CharT>::id [static]`

The facet id for ctype<char_type>

Definition at line 612 of file locale_facets.h.

The documentation for this class was generated from the following file:

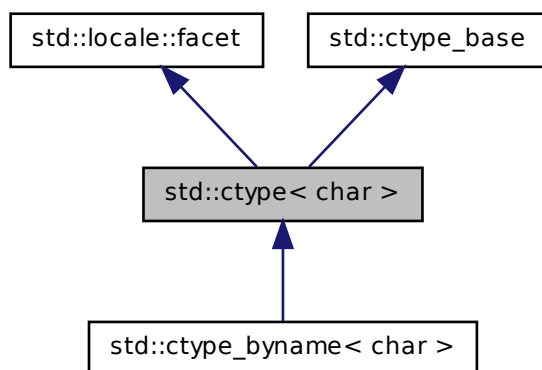
- [locale_facets.h](#)

5.420 `std::ctype< char >` Class Template Reference

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

Inheritance diagram for `std::ctype< char >`:



Public Types

- `typedef const int * __to_type`
- `typedef char char_type`
- `typedef unsigned short mask`

Public Member Functions

- `ctype` (`const mask *__table=0`, `bool __del=false`, `size_t __refs=0`)
- `ctype` (`__c_locale __cloc`, `const mask *__table=0`, `bool __del=false`, `size_t __refs=0`)
- `const char * is` (`const char *__lo`, `const char *__hi`, `mask *__vec`) `const`
- `bool is` (`mask __m`, `char __c`) `const`
- `char narrow` (`char_type __c`, `char __default`) `const`

- const [char_type](#) * [narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __to) const
- const char * [scan_is](#) (mask __m, const char * __lo, const char * __hi) const
- const char * [scan_not](#) (mask __m, const char * __lo, const char * __hi) const
- const mask * [table](#) () const throw ()
- [char_type](#) tolower ([char_type](#) __c) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) toupper ([char_type](#) __c) const
- [char_type](#) widen (char __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const

Static Public Member Functions

- static const mask * [classic_table](#) () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t [table_size](#)
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual [~ctype](#) ()
- virtual char [do_narrow](#) ([char_type](#) __c, char) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char, char * __dest) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#)) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#)) const

- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_widen](#) (char __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale [_M_c_locale_ctype](#)
- bool [_M_del](#)
- char [_M_narrow](#) [1+static_cast< unsigned char >(-1)]
- char [_M_narrow_ok](#)
- const mask * [_M_table](#)
- __to_type [_M_tolower](#)
- __to_type [_M_toupper](#)
- char [_M_widen](#) [1+static_cast< unsigned char >(-1)]
- char [_M_widen_ok](#)

Friends

- class [locale::_Impl](#)

5.420.1 Detailed Description

`template<> class std::ctype< char >`

The [ctype<char>](#) specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 673 of file `locale_facets.h`.

5.420.2 Member Typedef Documentation

5.420.2.1 `typedef char std::ctype< char >::char_type`

Typedef for the template parameter char.

Definition at line 678 of file locale_facets.h.

5.420.3 Constructor & Destructor Documentation

5.420.3.1 `std::ctype< char >::ctype (const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

table If non-zero, table is used as the per-char mask. Else [classic_table\(\)](#) is used.

del If true, passes ownership of table to this facet.

refs Passed to the base facet class.

5.420.3.2 `std::ctype< char >::ctype (__c_locale __cloc, const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

cloc Handle to C locale data.

table If non-zero, table is used as the per-char mask.

del If true, passes ownership of table to this facet.

refs Passed to the base facet class.

5.420.3.3 `virtual std::ctype< char >::~~ctype () [protected, virtual]`

Destructor.

This function deletes [table\(\)](#) if *del* was true in the constructor.

5.420.4 Member Function Documentation

5.420.4.1 `static const mask* std::ctype< char >::classic_table () throw ()`
`[static]`

Returns a pointer to the C locale mask table.

5.420.4.2 `virtual char std::ctype< char >::do_narrow (char_type __c, char`
`)const [inline, protected, virtual]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 1123 of file `locale_facets.h`.

5.420.4.3 `virtual const char_type* std::ctype< char >::do_narrow (const`
`char_type * __lo, const char_type * __hi, char, char * __dest)`
`const [inline, protected, virtual]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
dfault Char to use if conversion fails.
to Pointer to the destination array.

Returns

hi.

Definition at line 1149 of file locale_facets.h.

5.420.4.4 `virtual const char_type* std::ctype< char >::do_tolower (char_type
* __lo, const char_type * __hi) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.
hi Pointer to end of range.

Returns

hi.

5.420.4.5 `virtual char_type std::ctype< char >::do_tolower (char_type)
const [protected, virtual]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

5.420.4.6 `virtual const char_type* std::ctype< char >::do_toupper (char_type
* __lo, const char_type* __hi) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.420.4.7 `virtual char_type std::ctype< char >::do_toupper (char_type)
const [protected, virtual]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

5.420.4.8 `virtual char_type std::ctype< char >::do_widen (char __c) const [inline, protected, virtual]`

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 1074 of file `locale_facets.h`.

5.420.4.9 `virtual const char* std::ctype< char >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [inline, protected, virtual]`

Widen char array.

This function converts each char in the range `[lo,hi)` to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 1097 of file `locale_facets.h`.

5.420.4.10 `const char * std::ctype< char >::is (const char * __lo, const char * __hi, mask * __vec) const [inline]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 46 of file ctype_inline.h.

5.420.4.11 `bool std::ctype< char >::is (mask __m, char __c) const [inline]`

Test char classification.

This function compares the mask table[c] to *m*.

Parameters

c The char to compare the mask of.
m The mask to compare against.

Returns

True if *m* & table[c] is true, false otherwise.

Definition at line 41 of file ctype_inline.h.

5.420.4.12 `char std::ctype< char >::narrow (char_type __c, char __dfault) const [inline]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The char to convert.
- dfault* Char to return if conversion fails.

Returns

- The converted character.

Definition at line 922 of file `locale_facets.h`.

5.420.4.13 `const char_type* std::ctype<char>::narrow (const char_type *
__lo, const char_type * __hi, char __dfault, char * __to) const
[inline]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

- hi*.

Definition at line 955 of file `locale_facets.h`.

5.420.4.14 `const char * std::ctype< char >::scan_is (mask __m, const char * __lo, const char * __hi) const [inline]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char if found, else *hi*.

Definition at line 55 of file ctype_inline.h.

5.420.4.15 `const char * std::ctype< char >::scan_not (mask __m, const char * __lo, const char * __hi) const [inline]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 65 of file ctype_inline.h.

5.420.4.16 `const mask* std::ctype< char >::table () const throw () [inline]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic_table\(\)](#) if none was provided.

Definition at line 973 of file locale_facets.h.

5.420.4.17 `char_type std::ctype< char >::tolower (char_type __c) const [inline]`

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(c)`. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

Definition at line 827 of file locale_facets.h.

5.420.4.18 `const char_type* std::ctype< char >::tolower (char_type * __lo, const char_type * __hi) const [inline]`

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(lo, hi)`. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 844 of file locale_facets.h.

5.420.4.19 `const char_type* std::ctype< char >::toupper (char_type * __lo, const char_type * __hi) const [inline]`

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(lo, hi)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 811 of file locale_facets.h.

5.420.4.20 char_type std::ctype< char >::toupper (char_type __c) const [inline]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(c)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

Definition at line 794 of file locale_facets.h.

5.420.4.21 char_type std::ctype< char >::widen (char __c) const [inline]

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 864 of file locale_facets.h.

5.420.4.22 `const char* std::ctype< char >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 891 of file locale_facets.h.

5.420.5 Member Data Documentation

5.420.5.1 `locale::id std::ctype< char >::id [static]`

The facet id for `ctype<char>`

Definition at line 695 of file locale_facets.h.

5.420.5.2 `const size_t std::ctype< char >::table_size [static]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file locale_facets.h.

The documentation for this class was generated from the following files:

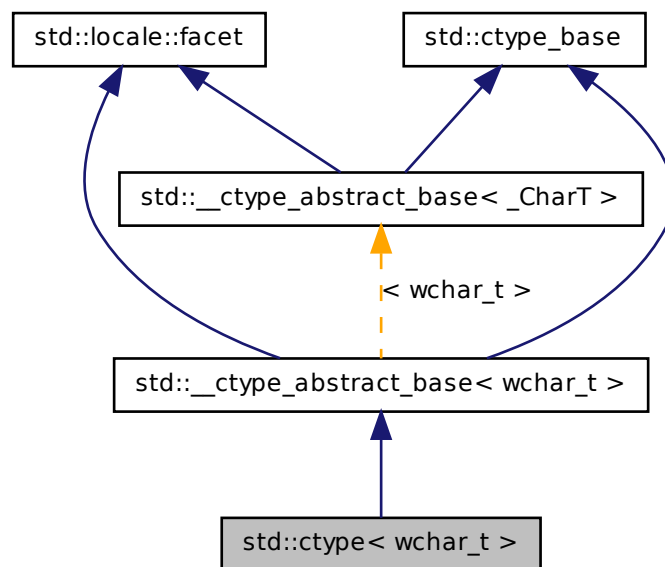
- [locale_facets.h](#)
- [ctype_inline.h](#)

5.421 std::ctype< wchar_t > Class Template Reference

The [ctype<wchar_t>](#) specialization.

This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.

Inheritance diagram for std::ctype< wchar_t >:



Public Types

- typedef const int * **__to_type**
- typedef wctype_t **__wmask_type**
- typedef wchar_t [char_type](#)
- typedef unsigned short **mask**

Public Member Functions

- [ctype](#) (size_t __refs=0)
- [ctype](#) (__c_locale __cloc, size_t __refs=0)
- const [char_type](#) * **is** (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- bool **is** (mask __m, [char_type](#) __c) const
- char **narrow** ([char_type](#) __c, char __dfault) const
- const [char_type](#) * **narrow** (const [char_type](#) * __lo, const [char_type](#) * __hi, char __dfault, char * __to) const
- const [char_type](#) * **scan_is** (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * **scan_not** (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) **tolower** ([char_type](#) __c) const
- const [char_type](#) * **tolower** ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) **toupper** ([char_type](#) __c) const
- const [char_type](#) * **toupper** ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const char * **widen** (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) **widen** (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual `~ctype` ()
- `__wmask_type _M_convert_to_wmask` (const mask __m) const throw ()
- void `_M_initialize_ctype` () throw ()
- virtual const `char_type * do_is` (const `char_type` *__lo, const `char_type` *__hi, mask *__vec) const
- virtual bool `do_is` (mask __m, `char_type` __c) const =0
- virtual const `char_type * do_is` (const `char_type` *__lo, const `char_type` *__hi, mask *__vec) const =0
- virtual bool `do_is` (mask __m, `char_type` __c) const
- virtual char `do_narrow` (`char_type`, char __dfault) const
- virtual char `do_narrow` (`char_type`, char __dfault) const =0
- virtual const `char_type * do_narrow` (const `char_type` *__lo, const `char_type` *__hi, char __dfault, char *__dest) const =0
- virtual const `char_type * do_narrow` (const `char_type` *__lo, const `char_type` *__hi, char __dfault, char *__dest) const
- virtual const `char_type * do_scan_is` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const =0
- virtual const `char_type * do_scan_is` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const
- virtual const `char_type * do_scan_not` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const
- virtual const `char_type * do_scan_not` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const =0
- virtual const `char_type * do_tolower` (`char_type` *__lo, const `char_type` *__hi) const
- virtual `char_type do_tolower` (`char_type`) const =0
- virtual const `char_type * do_tolower` (`char_type` *__lo, const `char_type` *__hi) const =0
- virtual `char_type do_tolower` (`char_type`) const
- virtual `char_type do_toupper` (`char_type`) const
- virtual const `char_type * do_toupper` (`char_type` *__lo, const `char_type` *__hi) const
- virtual `char_type do_toupper` (`char_type`) const =0
- virtual const `char_type * do_toupper` (`char_type` *__lo, const `char_type` *__hi) const =0
- virtual const char * `do_widen` (const char *__lo, const char *__hi, `char_type` *__dest) const =0
- virtual `char_type do_widen` (char) const
- virtual const char * `do_widen` (const char *__lo, const char *__hi, `char_type` *__dest) const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Protected Attributes

- mask `_M_bit` [16]
- `__c_locale _M_c_locale_ctype`
- char `_M_narrow` [128]
- bool `_M_narrow_ok`
- `wint_t _M_widen` [1+static_cast< unsigned char >(-1)]
- `__wmask_type _M_wmask` [16]

Friends

- class `locale::_Impl`

5.421.1 Detailed Description

`template<> class std::ctype< wchar_t >`

The [ctype<wchar_t>](#) specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well. [ctype<wchar_t>](#) inherits its public methods from [__ctype_abstract_base<wchar_t>](#).

Definition at line 1174 of file `locale_facets.h`.

5.421.2 Member Typedef Documentation

5.421.2.1 `typedef wchar_t std::ctype< wchar_t >::char_type`

Typedef for the template parameter `wchar_t`.

Reimplemented from [std::__ctype_abstract_base< wchar_t >](#).

Definition at line 1179 of file `locale_facets.h`.

5.421.3 Constructor & Destructor Documentation

5.421.3.1 std::ctype< wchar_t >::ctype (size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

5.421.3.2 std::ctype< wchar_t >::ctype (__c_locale __cloc, size_t __refs = 0) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

cloc Handle to C locale data.

refs Passed to the base facet class.

5.421.3.3 virtual std::ctype< wchar_t >::~~ctype () [protected, virtual]

Destructor.

5.421.4 Member Function Documentation

5.421.4.1 virtual bool std::ctype< wchar_t >::do_is (mask __m, char_type __c) const [protected, virtual]

Test wchar_t classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

c The wchar_t to find the mask of.

m The mask to compare against.

Returns

$(M \& m) \neq 0$.

5.421.4.2 `virtual bool std::__ctype_abstract_base< wchar_t >::do_is (mask
__m, char_type __c) const` [**protected**, **pure virtual**,
inherited]

Test char_type classification.

This function finds a mask M for c and compares it to mask m .

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

c The char_type to find the mask of.

m The mask to compare against.

Returns

$(M \& m) \neq 0$.

5.421.4.3 `virtual const char_type* std::ctype< wchar_t >::do_is (const
char_type * __lo, const char_type * __hi, mask * __vec) const`
[**protected**, **virtual**]

Return a mask array.

This function finds the mask for each `wchar_t` in the range $[lo,hi)$ and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi .

5.421.4.4 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_is(const char_type * __lo, const char_type * __hi, mask * __vec) const` `[protected, pure virtual, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

5.421.4.5 `virtual char std::__ctype_abstract_base< wchar_t >::do_narrow(char_type, char __dfault) const` `[protected, pure virtual, inherited]`

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

c The char_type to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

5.421.4.6 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, pure virtual, inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

5.421.4.7 `virtual char std::ctype< wchar_t >::do_narrow (char_type, char __dfault) const [protected, virtual]`

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<wchar_t>` facet, *c* will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `wchar_t` to convert.
- dfault* Char to return if conversion fails.

Returns

The converted `char`.

5.421.4.8 `virtual const char_type* std::ctype< wchar_t >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, virtual]`

Narrow wchar_t array to char array.

This virtual function converts each wchar_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar_t in the input that cannot be converted, *dfault* is used instead. For an undervived `ctype<wchar_t>` facet, the argument will be copied, casting each element to char.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

5.421.4.9 `virtual const char_type* std::ctype< wchar_t >::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find wchar_t matching mask.

This function searches for and returns the first wchar_t c in [lo,hi) for which `is(m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a matching wchar_t if found, else *hi*.

5.421.4.10 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

5.421.4.11 `virtual const char_type* std::ctype< wchar_t >::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find wchar_t not matching mask.

This function searches for and returns a pointer to the first wchar_t c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a non-matching wchar_t if found, else *hi*.

5.421.4.12 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_scan_not(mask __m, const char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *hi*.

5.421.4.13 `virtual char_type std::ctype< wchar_t >::do_tolower(char_type) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

c The wchar_t to convert.

Returns

The lowercase wchar_t if convertible, else *c*.

5.421.4.14 `virtual char_type std::__ctype_abstract_base< wchar_t >::do_tolower(char_type) const [protected, pure virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

5.421.4.15 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_tolower (char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.421.4.16 `virtual const char_type* std::ctype< wchar_t >::do_tolower (char_type * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

[`do_tolower\(\)`](#) is a hook for a derived facet to change the behavior of lowercasing. [`do_tolower\(\)`](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.421.4.17 `virtual char_type std::__ctype_abstract_base< wchar_t
>::do_toupper (char_type) const [protected, pure
virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

5.421.4.18 `virtual char_type std::ctype< wchar_t >::do_toupper (char_type
) const [protected, virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The `wchar_t` to convert.

Returns

The uppercase `wchar_t` if convertible, else *c*.

5.421.4.19 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_toupper (char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.421.4.20 `virtual const char_type* std::ctype< wchar_t >::do_toupper (char_type * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[`do_toupper\(\)`](#) is a hook for a derived facet to change the behavior of uppercasing. [`do_toupper\(\)`](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.421.4.21 `virtual char_type std::ctype< wchar_t >::do_widen (char) const [protected, virtual]`

Widen `char` to `wchar_t`.

This virtual function converts the char to wchar_t using the simplest reasonable transformation. For an underived [ctype<wchar_t>](#) facet, the argument will be cast to wchar_t.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv` for that.

Parameters

c The char to convert.

Returns

The converted wchar_t.

Implements [std::__ctype_abstract_base< wchar_t >](#).

5.421.4.22 `virtual const char* std::ctype< wchar_t >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, virtual]`

Widen char array to wchar_t array.

This function converts each char in the input to wchar_t using the simplest reasonable transformation. For an underived [ctype<wchar_t>](#) facet, the argument will be copied, casting each element to wchar_t.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv` for that.

Parameters

lo Pointer to start range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

5.421.4.23 `virtual const char* std::__ctype_abstract_base< wchar_t >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, pure virtual, inherited]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

lo Pointer to start range.
hi Pointer to end of range.
to Pointer to the destination array.

Returns

hi.

5.421.4.24 `const char_type* std::__ctype_abstract_base< wchar_t >::is (const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.421.4.25 `bool std::__ctype_abstract_base< wchar_t >::is (mask __m, char_type __c) const [inline, inherited]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m. It does so by returning the value of ctype<char_type>::do_is().

Parameters

- c* The char_type to compare the mask of.
- m* The mask to compare against.

Returns

(M & m) != 0.

Definition at line 161 of file locale_facets.h.

5.421.4.26 `const char_type* std::__ctype_abstract_base< wchar_t >::narrow (
const char_type * __lo, const char_type * __hi, char __dfault,
char * __to) const [inline, inherited]`

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

5.421.4.27 `char std::__ctype_abstract_base< wchar_t >::narrow (char_type
__c, char __dfault) const [inline, inherited]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. It does so by returning ctype<char_type>::do_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

c The char_type to convert.
dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 323 of file locale_facets.h.

5.421.4.28 `const char_type* std::__ctype_abstract_base< wchar_t >::scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [inline, inherited]`

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

m The mask to compare against.
lo Pointer to start of range.
hi Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

5.421.4.29 `const char_type* std::__ctype_abstract_base< wchar_t >::scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [inline, inherited]`

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

m The mask to compare against.
lo Pointer to first char in range.
hi Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.421.4.30 `const char_type* std::__ctype_abstract_base< wchar_t >::tolower (char_type * __lo, const char_type * __hi) const [inline, inherited]`

Convert array to lowercase.

This function converts each char_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>::do_toupper(lo, hi).

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file locale_facets.h.

5.421.4.31 `char_type std::__ctype_abstract_base< wchar_t >::tolower (char_type __c) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

c The char_type to convert.

Returns

The lowercase char_type if convertible, else *c*.

Definition at line 253 of file locale_facets.h.

5.421.4.32 `const char_type* std::__ctype_abstract_base< wchar_t >::toupper (char_type * __lo, const char_type * __hi) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.421.4.33 `char_type std::__ctype_abstract_base< wchar_t >::toupper (char_type __c) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.421.4.34 `char_type std::__ctype_abstract_base< wchar_t >::widen (char __c) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type.

Definition at line 285 of file locale_facets.h.

5.421.4.35 `const char* std::__ctype_abstract_base< wchar_t >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecv for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file locale_facets.h.

5.421.5 Member Data Documentation

5.421.5.1 `locale::id std::ctype< wchar_t >::id [static]`

The facet id for [ctype<wchar_t>](#)

Definition at line 1197 of file locale_facets.h.

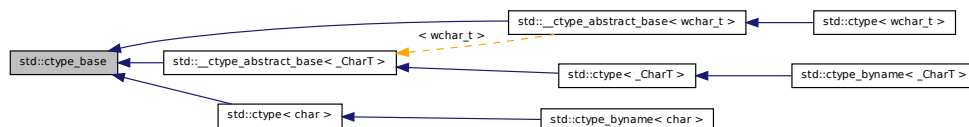
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.422 std::ctype_base Struct Reference

Base class for ctype.

Inheritance diagram for std::ctype_base:



Public Types

- typedef const int * **__to_type**
- typedef unsigned short **mask**

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

5.422.1 Detailed Description

Base class for ctype.

Definition at line 40 of file ctype_base.h.

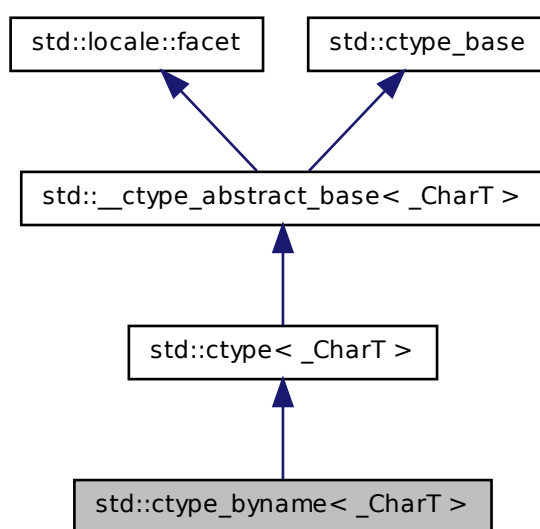
The documentation for this struct was generated from the following file:

- [ctype_base.h](#)

5.423 std::ctype_byname< _CharT > Class Template Reference

class [ctype_byname](#) [22.2.1.2].

Inheritance diagram for std::ctype_byname< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT [char_type](#)
- typedef [ctype](#)< _CharT >::mask **mask**

Public Member Functions

- **ctype_byname** (const char * __s, size_t __refs=0)
- bool **is** (mask __m, [char_type](#) __c) const
- const [char_type](#) * **is** (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __-vec) const

- const [char_type](#) * [narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __to) const
- char [narrow](#) ([char_type](#) __c, char __default) const
- const [char_type](#) * [scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __dest) const
- virtual char [do_narrow](#) ([char_type](#), char __default) const
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#) __c) const

- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#) __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const
- virtual [char_type](#) [do_widen](#) (char __c) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.423.1 Detailed Description

template<typename _CharT> class std::ctype_byname<_CharT>

class [ctype_byname](#) [22.2.1.2].

Definition at line 1466 of file locale_facets.h.

5.423.2 Member Typedef Documentation

5.423.2.1 **template<typename _CharT> typedef _CharT std::ctype<_CharT>::char_type [inherited]**

Typedef for the template parameter.

Reimplemented from [std::__ctype_abstract_base<_CharT>](#).

Definition at line 608 of file locale_facets.h.

5.423.3 Member Function Documentation

5.423.3.1 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const` `[protected, virtual, inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

Returns

hi.

Implements `std::__ctype_abstract_base<_CharT>`.

5.423.3.2 `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is (mask __m, char_type __c) const` `[protected, virtual, inherited]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

- c* The `char_type` to find the mask of.
- m* The mask to compare against.

Returns

$(M \& m) \neq 0$.

Implements `std::__ctype_abstract_base<_CharT>`.

5.423.3.3 `template<typename _CharT> virtual char std::ctype< _CharT >::do_narrow (char_type, char __dfault) const [protected, virtual, inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

Returns

The converted `char`.

Implements `std::__ctype_abstract_base< _CharT >`.

5.423.3.4 `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, virtual, inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *hi*.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.7 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower(char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each char_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.8 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower(char_type) const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char_type to convert.

Returns

The lowercase `char_type` if convertible, else `c`.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.9 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper(char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.10 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper(char_type) const [protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else `c`.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.11 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen(char) const [protected, virtual, inherited]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.12 `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen(const char * __lo, const char * __hi, char_type * __dest) const [protected, virtual, inherited]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.423.3.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is (mask __m, char_type __c) const [inline, inherited]`

Test char_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

- c* The char_type to compare the mask of.
- m* The mask to compare against.

Returns

(*M* & *m*) != 0.

Definition at line 161 of file locale_facets.h.

Referenced by `std::regex_traits<_Ch_type>::isctype()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.423.3.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is (const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.423.3.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow (char_type __c, char __dfault) const
[inline, inherited]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- c* The char_type to convert.
- dfault* Char to return if conversion fails.

Returns

The converted char.

Definition at line 323 of file locale_facets.h.

Referenced by std::time_put< _CharT, _OutIter >::put().

5.423.3.16 `template<typename _CharT> const char_type*
std::__ctype_abstract_base< _CharT >::narrow (const char_type
* __lo, const char_type * __hi, char __dfault, char * __to) const
[inline, inherited]`

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char_type>::do_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

```
5.423.3.17  template<typename _CharT> const char_type*
             std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,
             const char_type * __lo, const char_type * __hi ) const  [inline,
             inherited]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char_type>::do_scan_is\(\)](#).

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

```
5.423.3.18  template<typename _CharT> const char_type*
             std::__ctype_abstract_base<_CharT>::scan_not ( mask __m,
             const char_type * __lo, const char_type * __hi ) const  [inline,
             inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char_type>::do_scan_not\(\)](#).

Parameters

- m* The mask to compare against.
- lo* Pointer to first char in range.
- hi* Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.423.3.19 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::tolower (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file `locale_facets.h`.

5.423.3.20 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

5.423.3.21 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::toupper (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.423.3.22 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.423.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen (char __c) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type.

Definition at line 285 of file locale_facets.h.

Referenced by std::money_get<_CharT, _InIter>::do_get(), std::time_put<_CharT, _OutIter>::do_put(), std::money_put<_CharT, _OutIter>::do_put(), std::regex_traits<_Ch_type>::isctype(), and std::operator<<().

5.423.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file locale_facets.h.

5.423.4 Member Data Documentation

5.423.4.1 `template<typename _CharT> locale::id std::ctype<_CharT>::id [static, inherited]`

The facet id for ctype<char_type>

Definition at line 612 of file locale_facets.h.

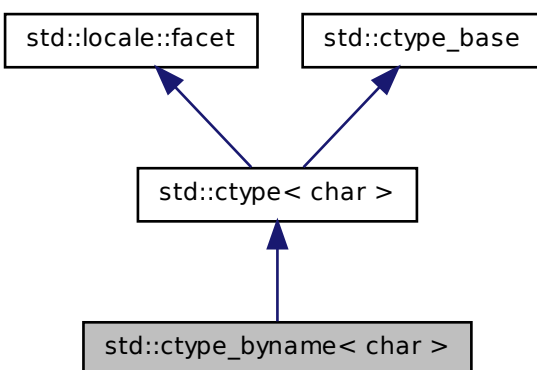
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.424 `std::ctype_byname< char >` Class Template Reference

22.2.1.4 Class `ctype_byname` specializations.

Inheritance diagram for `std::ctype_byname< char >`:



Public Types

- typedef const int * `__to_type`
- typedef char `char_type`
- typedef unsigned short `mask`

Public Member Functions

- `ctype_byname` (const char * __s, size_t __refs=0)
- bool `is` (mask __m, char __c) const
- const char * `is` (const char * __lo, const char * __hi, mask * __vec) const
- const `char_type` * `narrow` (const `char_type` * __lo, const `char_type` * __hi, char __default, char * __to) const
- char `narrow` (`char_type` __c, char __default) const
- const char * `scan_is` (mask __m, const char * __lo, const char * __hi) const
- const char * `scan_not` (mask __m, const char * __lo, const char * __hi) const

- const mask * [table](#) () const throw ()
- const [char_type](#) * [tolower](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const [char_type](#) * [toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- const char * [widen](#) (const char *__lo, const char *__hi, [char_type](#) *__to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Member Functions

- static const mask * [classic_table](#) () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t [table_size](#)
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual char [do_narrow](#) ([char_type](#) __c, char) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) *__lo, const [char_type](#) *__hi, char, char *__dest) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#)) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#)) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_widen](#) (char __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Protected Attributes

- `__c_locale _M_c_locale_ctype`
- `bool _M_del`
- `char _M_narrow [1+static_cast< unsigned char >(-1)]`
- `char _M_narrow_ok`
- `const mask * _M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- `char _M_widen [1+static_cast< unsigned char >(-1)]`
- `char _M_widen_ok`

Friends

- `class locale::_Impl`

5.424.1 Detailed Description

`template<> class std::ctype_byname< char >`

22.2.1.4 Class [ctype_byname](#) specializations.

Definition at line 1481 of file `locale_facets.h`.

5.424.2 Member Typedef Documentation

5.424.2.1 `typedef char std::ctype< char >::char_type` `[inherited]`

Typedef for the template parameter `char`.

Definition at line 678 of file `locale_facets.h`.

5.424.3 Member Function Documentation

5.424.3.1 `static const mask* std::ctype< char >::classic_table () throw ()
[static, inherited]`

Returns a pointer to the C locale mask table.

5.424.3.2 `virtual char std::ctype< char >::do_narrow (char_type __c, char
)const [inline, protected, virtual, inherited]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 1123 of file `locale_facets.h`.

5.424.3.3 `virtual const char_type* std::ctype< char >::do_narrow (const
char_type * __lo, const char_type * __hi, char, char * __dest)
const [inline, protected, virtual, inherited]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
dfault Char to use if conversion fails.
to Pointer to the destination array.

Returns

hi.

Definition at line 1149 of file locale_facets.h.

5.424.3.4 `virtual const char_type* std::ctype< char >::do_tolower (char_type
 * __lo, const char_type * __hi) const [protected, virtual,
 inherited]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.
hi Pointer to end of range.

Returns

hi.

5.424.3.5 `virtual char_type std::ctype< char >::do_tolower (char_type)
 const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

5.424.3.6 `virtual const char_type* std::ctype< char >::do_toupper (char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.424.3.7 `virtual char_type std::ctype< char >::do_toupper (char_type) const [protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

5.424.3.8 **virtual char_type std::ctype< char >::do_widen (char __c) const** [inline, protected, virtual, inherited]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be returned unchanged.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 1074 of file [locale_facets.h](#).

5.424.3.9 **virtual const char* std::ctype< char >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const** [inline, protected, virtual, inherited]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be copied unchanged.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 1097 of file [locale_facets.h](#).

5.424.3.10 `const char * std::ctype< char >::is (const char * __lo, const char * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

Returns

hi.

Definition at line 46 of file ctype_inline.h.

5.424.3.11 `bool std::ctype< char >::is (mask __m, char __c) const [inline, inherited]`

Test char classification.

This function compares the mask table[c] to *m*.

Parameters

- c* The char to compare the mask of.
- m* The mask to compare against.

Returns

True if *m* & table[c] is true, false otherwise.

Definition at line 41 of file ctype_inline.h.

5.424.3.12 `char std::ctype< char >::narrow (char_type __c, char __dfault) const [inline, inherited]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The char to convert.
- dfault* Char to return if conversion fails.

Returns

The converted character.

Definition at line 922 of file `locale_facets.h`.

5.424.3.13 `const char_type* std::ctype<char>::narrow (const char_type *
__lo, const char_type * __hi, char __dfault, char * __to) const
[inline, inherited]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 955 of file `locale_facets.h`.

5.424.3.14 `const char * std::ctype< char >::scan_is (mask __m, const char * __lo, const char * __hi) const [inline, inherited]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char if found, else *hi*.

Definition at line 55 of file ctype_inline.h.

5.424.3.15 `const char * std::ctype< char >::scan_not (mask __m, const char * __lo, const char * __hi) const [inline, inherited]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 65 of file ctype_inline.h.

5.424.3.16 `const mask* std::ctype< char >::table () const throw () [inline, inherited]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic_table\(\)](#) if none was provided.

Definition at line 973 of file locale_facets.h.

5.424.3.17 `char_type std::ctype< char >::tolower (char_type __c) const`
[inline, inherited]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(c)`. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

Definition at line 827 of file locale_facets.h.

5.424.3.18 `const char_type* std::ctype< char >::tolower (char_type * __lo,`
`const char_type * __hi) const` **[inline, inherited]**

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(lo, hi)`. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 844 of file locale_facets.h.

5.424.3.19 `const char_type* std::ctype< char >::toupper (char_type * __lo,`
`const char_type * __hi) const` **[inline, inherited]**

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(lo, hi)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 811 of file locale_facets.h.

5.424.3.20 char_type std::ctype< char >::toupper (char_type __c) const [inline, inherited]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(c)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

Definition at line 794 of file locale_facets.h.

5.424.3.21 char_type std::ctype< char >::widen (char __c) const [inline, inherited]

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 864 of file locale_facets.h.

5.424.3.22 `const char* std::ctype< char >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 891 of file locale_facets.h.

5.424.4 Member Data Documentation

5.424.4.1 `locale::id std::ctype< char >::id [static, inherited]`

The facet id for `ctype<char>`

Definition at line 695 of file locale_facets.h.

5.424.4.2 `const size_t std::ctype< char >::table_size [static, inherited]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.425 `std::decay< _Tp >` Class Template Reference

`decay`

Public Types

- `typedef __decay_selector< __remove_type >::__type type`

5.425.1 Detailed Description

`template<typename _Tp> class std::decay< _Tp >`

`decay`

Definition at line 428 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type_traits](#)

5.426 `std::decimal::decimal128` Class Reference

3.2.4 Class [decimal128](#).

Public Types

- `typedef float __decfloat128 __attribute__((mode(TD)))`

Public Member Functions

- `decimal128 (decimal32 d32)`
- `decimal128 (float __r)`
- `decimal128 (unsigned int __z)`
- `decimal128 (long __z)`

- **decimal128** (double __r)
- **decimal128** (unsigned long __z)
- **decimal128** (long long __z)
- **decimal128** ([decimal64](#) d64)
- **decimal128** (long double __r)
- **decimal128** (unsigned long long __z)
- [decimal128](#) (__decfloat128 __z)
- **decimal128** (int __z)
- __decfloat128 **__getval** (void)
- void **__setval** (__decfloat128 __x)
- [decimal128](#) & **operator*=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator*=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator*=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator*=** (int __rhs)
- [decimal128](#) & **operator*=** (unsigned int __rhs)
- [decimal128](#) & **operator*=** (long __rhs)
- [decimal128](#) & **operator*=** (unsigned long __rhs)
- [decimal128](#) & **operator*=** (unsigned long long __rhs)
- [decimal128](#) & **operator*=** (long long __rhs)
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=** (int __rhs)
- [decimal128](#) & **operator+=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator+=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator+=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator+=** (unsigned int __rhs)
- [decimal128](#) & **operator+=** (long __rhs)
- [decimal128](#) & **operator+=** (unsigned long __rhs)
- [decimal128](#) & **operator+=** (long long __rhs)
- [decimal128](#) & **operator+=** (unsigned long long __rhs)
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=** (long long __rhs)
- [decimal128](#) & **operator-=** (long __rhs)
- [decimal128](#) & **operator-=** (int __rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator-=** (unsigned long __rhs)
- [decimal128](#) & **operator-=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator-=** (unsigned long long __rhs)
- [decimal128](#) & **operator-=** (unsigned int __rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator/=** (long __rhs)

- [decimal128](#) & **operator/=** (long long __rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator/=** (unsigned int __rhs)
- [decimal128](#) & **operator/=** (unsigned long __rhs)
- [decimal128](#) & **operator/=** (unsigned long long __rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator/=** (int __rhs)

5.426.1 Detailed Description

3.2.4 Class [decimal128](#).

Definition at line 391 of file decimal.

5.426.2 Constructor & Destructor Documentation

5.426.2.1 `std::decimal::decimal128::decimal128 (__decfloat128 __z)`
[inline]

Conforming extension: Conversion from scalar decimal type.

Definition at line 416 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

5.427 std::decimal::decimal32 Class Reference

3.2.2 Class [decimal32](#).

Public Types

- typedef float __decfloat32 **__attribute__** ((mode(SD)))

Public Member Functions

- **decimal32** ([decimal64](#) __d64)
- **decimal32** (float __r)
- **decimal32** (unsigned int __z)
- **decimal32** (long __z)

- **decimal32** (double __r)
- **decimal32** (unsigned long __z)
- **decimal32** (long long __z)
- **decimal32** ([decimal128](#) __d128)
- **decimal32** (long double __r)
- **decimal32** (unsigned long long __z)
- [decimal32](#) (__decfloat32 __z)
- **decimal32** (int __z)
- __decfloat32 **__getval** (void)
- void **__setval** (__decfloat32 __x)
- [decimal32](#) & **operator*=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator*=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator*=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator*=** (int __rhs)
- [decimal32](#) & **operator*=** (unsigned int __rhs)
- [decimal32](#) & **operator*=** (long __rhs)
- [decimal32](#) & **operator*=** (unsigned long __rhs)
- [decimal32](#) & **operator*=** (unsigned long long __rhs)
- [decimal32](#) & **operator*=** (long long __rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** (int __rhs)
- [decimal32](#) & **operator+=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator+=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator+=** (unsigned int __rhs)
- [decimal32](#) & **operator+=** (long __rhs)
- [decimal32](#) & **operator+=** (unsigned long __rhs)
- [decimal32](#) & **operator+=** (long long __rhs)
- [decimal32](#) & **operator+=** (unsigned long long __rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** (long long __rhs)
- [decimal32](#) & **operator-=** (long __rhs)
- [decimal32](#) & **operator-=** (int __rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator-=** (unsigned long __rhs)
- [decimal32](#) & **operator-=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator-=** (unsigned long long __rhs)
- [decimal32](#) & **operator-=** (unsigned int __rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator/=** (long __rhs)

- [decimal32](#) & **operator/=** (long long __rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator/=** (unsigned int __rhs)
- [decimal32](#) & **operator/=** (unsigned long __rhs)
- [decimal32](#) & **operator/=** (unsigned long long __rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator/=** (int __rhs)

5.427.1 Detailed Description

3.2.2 Class [decimal32](#).

Definition at line 225 of file decimal.

5.427.2 Constructor & Destructor Documentation

5.427.2.1 `std::decimal::decimal32::decimal32 (__decfloat32 __z)`
`[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 249 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

5.428 std::decimal::decimal64 Class Reference

3.2.3 Class [decimal64](#).

Public Types

- typedef float __decfloat64 **__attribute__** ((mode(DD)))

Public Member Functions

- **decimal64** ([decimal32](#) d32)
- **decimal64** (float __r)
- **decimal64** (unsigned int __z)
- **decimal64** (long __z)

- **decimal64** (double __r)
- **decimal64** (unsigned long __z)
- **decimal64** (long long __z)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (long double __r)
- **decimal64** (unsigned long long __z)
- [decimal64](#) (__decfloat64 __z)
- **decimal64** (int __z)
- __decfloat64 **__getval** (void)
- void **__setval** (__decfloat64 __x)
- [decimal64](#) & **operator*=** ([decimal32](#) __rhs)
- [decimal64](#) & **operator*=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator*=** ([decimal128](#) __rhs)
- [decimal64](#) & **operator*=** (int __rhs)
- [decimal64](#) & **operator*=** (unsigned int __rhs)
- [decimal64](#) & **operator*=** (long __rhs)
- [decimal64](#) & **operator*=** (unsigned long __rhs)
- [decimal64](#) & **operator*=** (unsigned long long __rhs)
- [decimal64](#) & **operator*=** (long long __rhs)
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=** (int __rhs)
- [decimal64](#) & **operator+=** ([decimal32](#) __rhs)
- [decimal64](#) & **operator+=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator+=** ([decimal128](#) __rhs)
- [decimal64](#) & **operator+=** (unsigned int __rhs)
- [decimal64](#) & **operator+=** (long __rhs)
- [decimal64](#) & **operator+=** (unsigned long __rhs)
- [decimal64](#) & **operator+=** (long long __rhs)
- [decimal64](#) & **operator+=** (unsigned long long __rhs)
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=** (long long __rhs)
- [decimal64](#) & **operator-=** (long __rhs)
- [decimal64](#) & **operator-=** (int __rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) __rhs)
- [decimal64](#) & **operator-=** (unsigned long __rhs)
- [decimal64](#) & **operator-=** ([decimal128](#) __rhs)
- [decimal64](#) & **operator-=** (unsigned long long __rhs)
- [decimal64](#) & **operator-=** (unsigned int __rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator/=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator/=** (long __rhs)

- [decimal64](#) & `operator/=` (long long __rhs)
- [decimal64](#) & `operator/=` ([decimal32](#) __rhs)
- [decimal64](#) & `operator/=` (unsigned int __rhs)
- [decimal64](#) & `operator/=` (unsigned long __rhs)
- [decimal64](#) & `operator/=` (unsigned long long __rhs)
- [decimal64](#) & `operator/=` ([decimal128](#) __rhs)
- [decimal64](#) & `operator/=` (int __rhs)

5.428.1 Detailed Description

3.2.3 Class [decimal64](#).

Definition at line 308 of file decimal.

5.428.2 Constructor & Destructor Documentation

5.428.2.1 `std::decimal::decimal64::decimal64 (__decfloat64 __z)`
[`inline`]

Conforming extension: Conversion from scalar decimal type.

Definition at line 332 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

5.429 `std::default_delete<_Tp>` Struct Template Reference

Primary template, [default_delete](#).

Public Member Functions

- `template<typename _Up>`
`default_delete` (const [default_delete](#)<_Up> &)
- `void operator() (_Tp *__ptr) const`

5.429.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp >`

Primary template, [default_delete](#).

Definition at line 48 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

5.430 `std::default_delete< _Tp[]>` Struct Template Reference

Specialization, [default_delete](#).

Public Member Functions

- `void operator() (_Tp *__ptr) const`

5.430.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp[] >`

Specialization, [default_delete](#).

Definition at line 68 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

5.431 `std::defer_lock_t` Struct Reference

Do not acquire ownership of the mutex.

5.431.1 Detailed Description

Do not acquire ownership of the mutex.

Definition at line 375 of file `mutex`.

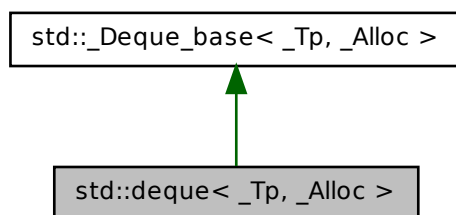
The documentation for this struct was generated from the following file:

- [mutex](#)

5.432 `std::deque< _Tp, _Alloc >` Class Template Reference

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Inheritance diagram for `std::deque< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- [deque](#) ()
- [deque](#) (const allocator_type &__a)
- [deque](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
[deque](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- [deque](#) (const [deque](#) &__x)
- [deque](#) (size_type __n)
- [deque](#) ([deque](#) &&__x)
- [deque](#) (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- [~deque](#) ()
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename _InputIterator >
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (initializer_list< value_type > __l)
- reference [at](#) (size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- template<typename... _Args>
[iterator](#) [emplace](#) (iterator __position, _Args &&...__args)
- template<typename... _Args>
void [emplace_back](#) (_Args &&...__args)
- template<typename... _Args>
void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- iterator [erase](#) (iterator __position)
- iterator [erase](#) (iterator __first, iterator __last)
- const_reference [front](#) () const
- reference [front](#) ()

- allocator_type [get_allocator](#) () const
- iterator [insert](#) (iterator __position, value_type &&__x)
- void [insert](#) (iterator __p, initializer_list< value_type > __l)
- template<typename _InputIterator >
void [insert](#) (iterator __position, _InputIterator __first, _InputIterator __last)
- void [insert](#) (iterator __position, size_type __n, const value_type &__x)
- iterator [insert](#) (iterator __position, const value_type &__x)
- size_type [max_size](#) () const
- deque & [operator=](#) (initializer_list< value_type > __l)
- deque & [operator=](#) (const deque &__x)
- deque & [operator=](#) (deque &&__x)
- reference [operator\[\]](#) (size_type __n)
- const_reference [operator\[\]](#) (size_type __n) const
- void [pop_back](#) ()
- void [pop_front](#) ()
- void [push_back](#) (value_type &&__x)
- void [push_back](#) (const value_type &__x)
- void [push_front](#) (value_type &&__x)
- void [push_front](#) (const value_type &__x)
- reverse_iterator [rbegin](#) ()
- const_reverse_iterator [rbegin](#) () const
- reverse_iterator [rend](#) ()
- const_reverse_iterator [rend](#) () const
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const value_type &__x)
- void [shrink_to_fit](#) ()
- size_type [size](#) () const
- void [swap](#) (deque &__x)

Protected Types

- enum { **_S_initial_map_size** }
- typedef _Alloc::template rebind< _Tp * >::other **_Map_alloc_type**
- typedef pointer * **_Map_pointer**

Protected Member Functions

- _Tp ** **_M_allocate_map** (size_t __n)
- _Tp * **_M_allocate_node** ()
- template<typename _ForwardIterator >
void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last,
[std::forward_iterator_tag](#))

- `template<typename _InputIterator >`
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _Integer >`
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Tp **__nstart, _Tp **__nfinish)`
- `void _M_deallocate_map (_Tp **__p, size_t __n)`
- `void _M_deallocate_node (_Tp *__p)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator<_Tp> &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp **__nstart, _Tp **__nfinish)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Map_alloc_type _M_get_map_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `template<typename _Integer >`
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_initialize_map (size_t)`
- `template<typename... _Args>`
`iterator _M_insert_aux (iterator __pos, _Args &&...__args)`
- `template<typename _ForwardIterator >`
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type &__x)`
- `template<typename _Integer >`
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`

- template<typename _InputIterator >
void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void **_M_range_check** (size_type __n) const
- template<typename _ForwardIterator >
void **_M_range_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator >
void **_M_range_insert_aux** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator >
void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename... _Args>
void **_M_push_back_aux** (_Args &&...__args)
- template<typename... _Args>
void **_M_push_front_aux** (_Args &&...__args)
- void **_M_pop_back_aux** ()
- void **_M_pop_front_aux** ()
- iterator **_M_reserve_elements_at_front** (size_type __n)
- iterator **_M_reserve_elements_at_back** (size_type __n)
- void **_M_new_elements_at_front** (size_type __new_elements)
- void **_M_new_elements_at_back** (size_type __new_elements)
- void **_M_reserve_map_at_back** (size_type __nodes_to_add=1)
- void **_M_reserve_map_at_front** (size_type __nodes_to_add=1)
- void **_M_reallocate_map** (size_type __nodes_to_add, bool __add_at_front)

Static Protected Member Functions

- static size_t **_S_buffer_size** ()

Protected Attributes

- **_Deque_impl** **_M_impl**

5.432.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::deque<_Tp, _Alloc>
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a deque<Tp> manages memory. Each deque has 4 members:

- Tp** _M_map
- size_t _M_map_size
- iterator _M_start, _M_finish

map_size is at least 8. map is an array of map_size pointers-to-. (The name map has nothing to do with the [std::map](#) class, and **nodes** should not be confused with [std::list](#)'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator i:
 - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
 - i.first == *(i.node) (This points to the node (first Tp element).)
 - i.last == i.first + node_size

- i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map_size).
- A pointer in the range [map, map + map_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, _Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 717 of file stl_deque.h.

5.432.2 Constructor & Destructor Documentation

5.432.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque () [inline]`

Default constructor creates no elements.

Definition at line 769 of file stl_deque.h.

5.432.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (const allocator_type & __a)
[inline, explicit]`

Creates a deque with no elements.

Parameters

a An allocator object.

Definition at line 777 of file `stl_deque.h`.

5.432.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (size_type __n) [inline,
explicit]`

Creates a deque with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor fills the deque with *n* default constructed elements.

Definition at line 789 of file `stl_deque.h`.

5.432.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (size_type __n, const value_type
& __value, const allocator_type & __a = allocator_type())
[inline]`

Creates a deque with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

a An allocator.

This constructor fills the deque with *n* copies of *value*.

Definition at line 801 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_fill_initialize()`.

5.432.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (const deque<_Tp, _Alloc> &
__x) [inline]`

Deque copy constructor.

Parameters

x A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocation object used by *x*.

Definition at line 828 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin(), and std::deque< _Tp, _Alloc >::end().

5.432.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (deque< _Tp, _Alloc > && __x
) [inline]`

Deque move constructor.

Parameters

x A deque of identical element and allocator types.

The newly-created deque contains the exact contents of *x*. The contents of *x* are a valid, but unspecified deque.

Definition at line 842 of file stl_deque.h.

5.432.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (initializer_list< value_type
> __l, const allocator_type & __a = allocator_type())
[inline]`

Builds a deque from an initializer list.

Parameters

l An [initializer_list](#).

a An allocator object.

Create a deque consisting of copies of the elements in the [initializer_list](#) *l*.

This will call the element type's copy constructor *N* times (where *N* is *l.size()*) and do no memory reallocation.

Definition at line 856 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::M_range_initialize().

5.432.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename _InputIterator > std::deque< _Tp, _Alloc
 >::deque (_InputIterator __first, _InputIterator __last, const
 allocator_type & __a = allocator_type()) [inline]`

Builds a deque from a range.

Parameters

first An input iterator.

last An input iterator.

a An allocator object.

Create a deque consisting of copies of the elements from [first, last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 881 of file stl_deque.h.

5.432.2.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::deque< _Tp, _Alloc >::~~deque () [inline]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 895 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::begin()`, and `std::deque< _Tp, _Alloc >::end()`.

5.432.3 Member Function Documentation

5.432.3.1 `template<typename _Tp , typename _Alloc > void
 deque::_M_fill_initialize (const value_type & __value)
 [protected]`

Fills the deque with copies of value.

Parameters

value Initial value.

Returns

Nothing.

Precondition

_M_start and _M_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 331 of file deque.tcc.

References std::_Destroy().

Referenced by std::deque< _Tp, _Alloc >::deque().

5.432.3.2 `template<typename _Tp , typename _Alloc > void
std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t
__num_elements) [protected, inherited]`

Layout storage.

Parameters

num_elements The count of T's for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 572 of file stl_deque.h.

References std::max().

Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize().

5.432.3.3 `template<typename _Tp , typename _Alloc > void
deque::_M_new_elements_at_back (size_type __new_elements)
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 825 of file deque.tcc.

References std::deque< _Tp, _Alloc >::_M_reserve_map_at_back(), std::deque< _Tp, _Alloc >::max_size(), and std::deque< _Tp, _Alloc >::size().

Referenced by std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back().

5.432.3.4 `template<typename _Tp , typename _Alloc > void
deque::_M_new_elements_at_front (size_type __new_elements)
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 800 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`, `std::deque< _Tp, _Alloc >::max_size()`, and `std::deque< _Tp, _Alloc >::size()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front()`.

5.432.3.5 `template<typename _Tp , typename _Alloc > void
deque::_M_pop_back_aux () [protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 481 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::pop_back()`.

5.432.3.6 `template<typename _Tp , typename _Alloc > void
deque::_M_pop_front_aux () [protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 496 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::pop_front()`.

5.432.3.7 `template<typename _Tp , typename _Alloc > template<typename...
_Args> void deque::_M_push_back_aux (_Args &&... __args)
[protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 415 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::push_back()`.

5.432.3.8 `template<typename _Tp , typename _Alloc > template<typename...
_Args> void deque::_M_push_front_aux (_Args &&... __args)
[protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 449 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::push_front().

5.432.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_range_check (size_type __n)
const [inline, protected]`

Safety check used only from [at\(\)](#).

Definition at line 1235 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::size().

Referenced by std::deque< _Tp, _Alloc >::at().

5.432.3.10 `template<typename _Tp , typename _Alloc > template<typename
_InputIterator > void deque::_M_range_initialize (_InputIterator
__first, _InputIterator __last, std::input_iterator_tag)
[protected]`

Fills the deque with whatever is in [first,last).

Parameters

first An input iterator.

last An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.

Definition at line 357 of file deque.tcc.

References std::_Deque_base< _Tp, _Alloc >::_M_initialize_map(), std::deque< _Tp, _Alloc >::clear(), and std::deque< _Tp, _Alloc >::push_back().

Referenced by std::deque< _Tp, _Alloc >::deque().

5.432.3.11 `template<typename _Tp , typename _Alloc > template<typename
_ForwardIterator > void deque::_M_range_initialize (
_ForwardIterator __first, _ForwardIterator __last,
std::forward_iterator_tag) [protected]`

Fills the deque with whatever is in [first,last).

Parameters

first An input iterator.

last An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 377 of file `deque.tcc`.

References `std::_Destroy()`, `std::_Deque_base< _Tp, _Alloc >::_M_initialize_map()`, `std::advance()`, and `std::distance()`.

5.432.3.12 `template<typename _Tp, typename _Alloc > void
deque::_M_reallocate_map (size_type __nodes_to_add, bool
__add_at_front) [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 850 of file `deque.tcc`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_map_at_back()`, and `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`.

5.432.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back
(size_type __n) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1856 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

5.432.3.14 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front
(size_type __n) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1846 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_new_elements_at_front().

5.432.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (
size_type __nodes_to_add = 1) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 1882 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_reallocate_map().

Referenced by std::deque< _Tp, _Alloc >::_M_new_elements_at_back().

5.432.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (
size_type __nodes_to_add = 1) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 1890 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_reallocate_map().

Referenced by std::deque< _Tp, _Alloc >::_M_new_elements_at_front().

5.432.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (size_type __n, const
value_type & __val) [inline]`

Assigns a given value to a deque.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 956 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::operator=()`.

```
5.432.3.18  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               template<typename _InputIterator > void std::deque<_Tp, _Alloc
               >::assign ( _InputIterator __first, _InputIterator __last )
               [inline]
```

Assigns a range to a deque.

Parameters

first An input iterator.

last An input iterator.

This function fills a deque with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 973 of file `stl_deque.h`.

```
5.432.3.19  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               void std::deque<_Tp, _Alloc>::assign ( initializer_list<
               value_type > __l ) [inline]
```

Assigns an initializer list to a deque.

Parameters

l An [initializer_list](#).

This function fills a deque with copies of the elements in the [initializer_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 992 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::assign()`.

Referenced by `std::deque<_Tp, _Alloc>::assign()`.

```
5.432.3.20  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               reference std::deque<_Tp, _Alloc>::at ( size_type __n )
               [inline]
```

Provides access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws *out_of_range* if the check fails.

Definition at line 1254 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_check().

```
5.432.3.21  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               const_reference std::deque< _Tp, _Alloc >::at ( size_type __n )
               const [inline]
```

Provides access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws *out_of_range* if the check fails.

Definition at line 1272 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_check().

```
5.432.3.22  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               const_reference std::deque< _Tp, _Alloc >::back ( ) const
               [inline]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1311 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::end()`.

5.432.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back () [inline]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1299 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::end()`.

5.432.3.24 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin () [inline]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1007 of file stl_deque.h.

Referenced by `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::deque()`, `std::deque< _Tp, _Alloc >::erase()`, `std::deque< _Tp, _Alloc >::front()`, `std::deque< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::deque< _Tp, _Alloc >::~~deque()`.

5.432.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1015 of file stl_deque.h.

5.432.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1078 of file stl_deque.h.

5.432.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1087 of file stl_deque.h.

5.432.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::clear () [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1577 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin().

Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize(), std::deque< _Tp, _Alloc >::erase(), and std::deque< _Tp, _Alloc >::operator=().

5.432.3.29 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin ()
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1096 of file stl_deque.h.

5.432.3.30 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1105 of file stl_deque.h.

5.432.3.31 `template<typename _Tp, typename _Alloc > template<typename...
_Args> deque< _Tp, _Alloc >::iterator deque::emplace (iterator
__position, _Args &&... __args)`

Inserts an object in deque before specified iterator.

Parameters

position An iterator into the deque.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location.`

Definition at line 172 of file `deque.tcc`.

References `std::deque<_Tp, _Alloc >::push_back()`, and `std::deque<_Tp, _Alloc >::push_front()`.

Referenced by `std::deque<_Tp, _Alloc >::insert()`.

5.432.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque<_Tp, _Alloc >::empty () const [inline]`

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

Definition at line 1198 of file `stl_deque.h`.

5.432.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1024 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc >::back()`, `std::deque<_Tp, _Alloc >::deque()`, `std::deque<_Tp, _Alloc >::erase()`, `std::deque<_Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::deque<_Tp, _Alloc >::~~deque()`.

5.432.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque<_Tp, _Alloc >::end () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1033 of file `stl_deque.h`.

5.432.3.35 `template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator __position)`

Remove element at given position.

Parameters

position Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 194 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `std::deque< _Tp, _Alloc >::pop_back()`, `std::deque< _Tp, _Alloc >::pop_front()`, and `std::deque< _Tp, _Alloc >::size()`.

5.432.3.36 `template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator __first, iterator __last)`

Remove a range of elements.

Parameters

first Iterator pointing to the first element to be erased.

last Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 217 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::end()`, and `std::deque< _Tp, _Alloc >::size()`.

5.432.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::front () [inline]`

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1283 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

5.432.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1291 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

5.432.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque<_Tp, _Alloc>::get_allocator () const
[inline]`

Get a copy of the memory allocation object.

Reimplemented from `std::_Deque_base<_Tp, _Alloc>`.

Definition at line 998 of file `stl_deque.h`.

5.432.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::deque<_Tp,
_Alloc>::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]`

Inserts a range into the deque.

Parameters

position An iterator into the deque.

first An input iterator.

last An input iterator.

This function will insert copies of the data in the range `[first,last)` into the deque before the location specified by *pos*. This is known as *range insert*.

Definition at line 1504 of file `stl_deque.h`.

5.432.3.41 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::insert (iterator __position, const value_type & __x)`

Inserts given value into deque before specified iterator.

Parameters

position An iterator into the deque.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 149 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::push_back()`, and `std::deque< _Tp, _Alloc >::push_front()`.

Referenced by `std::deque< _Tp, _Alloc >::operator=()`, and `std::deque< _Tp, _Alloc >::resize()`.

5.432.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert (iterator __p, initializer_list< value_type > __l) [inline]`

Inserts an initializer list into the deque.

Parameters

p An iterator into the deque.

l An [initializer_list](#).

This function will insert copies of the data in the [initializer_list](#) *l* into the deque before the location specified by *p*. This is known as *list insert*.

Definition at line 1475 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::insert()`.

Referenced by `std::deque< _Tp, _Alloc >::insert()`.

5.432.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::insert (iterator __position, value_type && __x) [inline]`

Inserts given rvalue into deque before specified iterator.

Parameters

position An iterator into the deque.
x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1462 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::emplace().

5.432.3.44 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::deque< _Tp, _Alloc >::insert (iterator __position,
 size_type __n, const value_type & __x) [inline]`

Inserts a number of copies of given data into the deque.

Parameters

position An iterator into the deque.
n Number of elements to be inserted.
x Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Definition at line 1489 of file stl_deque.h.

5.432.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 size_type std::deque< _Tp, _Alloc >::max_size () const
 [inline]`

Returns the [size\(\)](#) of the largest possible deque.

Definition at line 1117 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::M_new_elements_at_back(), and std::deque< _Tp, _Alloc >::M_new_elements_at_front().

5.432.3.46 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc
 > & deque::operator= (const deque< _Tp, _Alloc > & __x)`

Deque assignment operator.

Parameters

x A deque of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 93 of file deque.tcc.

References std::deque< _Tp, _Alloc >::begin(), std::deque< _Tp, _Alloc >::end(), std::deque< _Tp, _Alloc >::insert(), and std::deque< _Tp, _Alloc >::size().

5.432.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (initializer_list<
value_type > __l) [inline]`

Assigns an initializer list to a deque.

Parameters

l An [initializer_list](#).

This function fills a deque with copies of the elements in the [initializer_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 938 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::assign().

5.432.3.48 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (deque< _Tp,
_Alloc > && __x) [inline]`

Deque move assignment operator.

Parameters

x A deque of identical element and allocator types.

The contents of *x* are moved into this deque (without copying). *x* is a valid, but unspecified deque.

Definition at line 917 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::clear(), and std::deque< _Tp, _Alloc >::swap().

5.432.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::operator[] (size_type __n)
[inline]`

Subscript access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 1214 of file `stl_deque.h`.

5.432.3.50 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::operator[] (size_type
__n) const [inline]`

Subscript access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 1229 of file `stl_deque.h`.

5.432.3.51 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::pop_back () [inline]`

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before [pop_back\(\)](#) is called.

Definition at line 1412 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_pop_back_aux().

Referenced by std::deque< _Tp, _Alloc >::erase().

5.432.3.52 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop_front\(\)](#) is called.

Definition at line 1391 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_pop_front_aux().

Referenced by std::deque< _Tp, _Alloc >::erase().

5.432.3.53 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (const value_type &
__x) [inline]`

Add data to the end of the deque.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1360 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_push_back_aux().

Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize(), std::deque< _Tp, _Alloc >::emplace(), and std::deque< _Tp, _Alloc >::insert().

5.432.3.54 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (const value_type &
__x) [inline]`

Add data to the front of the deque.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1329 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_push_front_aux()`.

Referenced by `std::deque< _Tp, _Alloc >::emplace()`, and `std::deque< _Tp, _Alloc >::insert()`.

5.432.3.55 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1051 of file `stl_deque.h`.

5.432.3.56 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin ()
[inline]`

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1042 of file `stl_deque.h`.

5.432.3.57 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1060 of file `stl_deque.h`.

5.432.3.58 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1069 of file stl_deque.h.

```
5.432.3.59  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            void std::deque< _Tp, _Alloc >::resize ( size_type __new_size )
            [inline]
```

Resizes the deque to the specified number of elements.

Parameters

new_size Number of elements the deque should contain.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1131 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::size().

```
5.432.3.60  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            void std::deque< _Tp, _Alloc >::resize ( size_type __new_size,
            const value_type & __x ) [inline]
```

Resizes the deque to the specified number of elements.

Parameters

new_size Number of elements the deque should contain.

x Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1153 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::insert(), and std::deque< _Tp, _Alloc >::size().

```
5.432.3.61  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline]
```

A non-binding request to reduce memory use.

Definition at line 1189 of file stl_deque.h.

5.432.3.62 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque<_Tp, _Alloc>::size () const [inline]`

Returns the number of elements in the deque.

Definition at line 1112 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::_M_new_elements_at_back()`, `std::deque<_Tp, _Alloc>::_M_new_elements_at_front()`, `std::deque<_Tp, _Alloc>::_M_range_check()`, `std::deque<_Tp, _Alloc>::erase()`, `std::deque<_Tp, _Alloc>::operator=()`, `std::operator==()`, and `std::deque<_Tp, _Alloc>::resize()`.

5.432.3.63 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::swap (deque<_Tp, _Alloc> &
__x) [inline]`

Swaps data with another deque.

Parameters

x A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Definition at line 1557 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::operator=()`, and `std::swap()`.

The documentation for this class was generated from the following files:

- [stl_deque.h](#)
- [deque.tcc](#)

5.433 `std::discard_block_engine<
RandomNumberEngine, __p, __r > Class
Template Reference`

Public Types

- `typedef RandomNumberEngine::result_type` [result_type](#)

Public Member Functions

- [discard_block_engine](#) ()
- [discard_block_engine](#) (const [_RandomNumberEngine](#) &__rne)
- [discard_block_engine](#) (result_type __s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>`
[discard_block_engine](#) (_Sseq &__q)
- [discard_block_engine](#) ([_RandomNumberEngine](#) &&__rne)
- const [_RandomNumberEngine](#) & [base](#) () const
- void [discard](#) (unsigned long long __z)
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- [result_type](#) [operator\(\)](#) ()
- void [seed](#) (result_type __s)
- `template<typename _Sseq >`
 void [seed](#) (_Sseq &__q)
- void [seed](#) ()

Static Public Attributes

- static const size_t [block_size](#)
- static const size_t [used_block](#)

Friends

- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >`
[std::basic_ostream](#)<_CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)<_CharT, _Traits > &, const [std::discard_block_engine](#)<_RandomNumberEngine1, __p1, __r1 > &)
- bool [operator==](#) (const [discard_block_engine](#) &__lhs, const [discard_block_engine](#) &__rhs)
- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >`
[std::basic_istream](#)<_CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)<_CharT, _Traits > &, [std::discard_block_engine](#)<_RandomNumberEngine1, __p1, __r1 > &)

5.433.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r> class
std::discard_block_engine< _RandomNumberEngine, __p, __r >
```

Produces random numbers from some base engine by discarding blocks of data.

0 <= __r <= __p

Definition at line 784 of file random.h.

5.433.2 Member Typedef Documentation

```
5.433.2.1 template<typename _RandomNumberEngine, size_t __p,
size_t __r> typedef _RandomNumberEngine::result_type
std::discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

Definition at line 791 of file random.h.

5.433.3 Constructor & Destructor Documentation

```
5.433.3.1 template<typename _RandomNumberEngine, size_t __p, size_t
__r> std::discard_block_engine< _RandomNumberEngine, __p, __r
>::discard_block_engine ( ) [inline]
```

Constructs a default discard_block_engine engine.

The underlying engine is default constructed as well.

Definition at line 802 of file random.h.

```
5.433.3.2 template<typename _RandomNumberEngine, size_t __p, size_t
__r> std::discard_block_engine< _RandomNumberEngine, __p, __r
>::discard_block_engine ( const _RandomNumberEngine & __rne )
[inline, explicit]
```

Copy constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

rne An existing (base class) engine object.

5.433 std::discard_block_engine<_RandomNumberEngine, __p, __r> Class Template Reference 2401

Definition at line 812 of file random.h.

5.433.3.3 `template<typename _RandomNumberEngine, size_t __p, size_t
__r> std::discard_block_engine<_RandomNumberEngine, __p, __r
>::discard_block_engine (_RandomNumberEngine && __rne)
[inline, explicit]`

Move constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

rne An existing (base class) engine object.

Definition at line 822 of file random.h.

5.433.3.4 `template<typename _RandomNumberEngine, size_t __p, size_t
__r> std::discard_block_engine<_RandomNumberEngine, __p,
__r>::discard_block_engine (result_type __s) [inline,
explicit]`

Seed constructs a discard_block_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

s A seed value for the base class engine.

Definition at line 832 of file random.h.

5.433.3.5 `template<typename _RandomNumberEngine, size_t __p,
size_t __r> template<typename _Sseq, typename = typename
std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value
&& !std::is_same<_Sseq, _RandomNumberEngine>::value>
::type> std::discard_block_engine<_RandomNumberEngine,
__p, __r>::discard_block_engine (_Sseq & __q) [inline,
explicit]`

Generator construct a discard_block_engine engine.

Parameters

q A seed sequence.

Definition at line 845 of file random.h.

5.433.4 Member Function Documentation

5.433.4.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> const _RandomNumberEngine& std::discard_block_engine<_RandomNumberEngine, __p, __r>::base () const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 889 of file random.h.

5.433.4.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard (unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 916 of file random.h.

5.433.4.3 `template<typename _RandomNumberEngine, size_t __p, size_t __r> result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::max () const [inline]`

Gets the maximum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 907 of file random.h.

5.433.4.4 `template<typename _RandomNumberEngine, size_t __p, size_t __r> result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::min () const [inline]`

Gets the minimum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 898 of file random.h.

5.433.4.5 `template<typename _RandomNumberEngine, size_t __p, size_t
__r> discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type std::discard_block_engine< _RandomNumberEngine,
__p, __r >::operator() ()`

Gets the next value in the generated random number sequence.

Definition at line 662 of file random.tcc.

5.433.4.6 `template<typename _RandomNumberEngine, size_t __p, size_t
__r> template<typename _Sseq > void std::discard_block_engine<
_RandomNumberEngine, __p, __r >::seed (_Sseq & __q)
[inline]`

Reseeds the discard_block_engine object with the given seed sequence.

Parameters

`__q` A seed generator function.

Definition at line 878 of file random.h.

5.433.4.7 `template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r
>::seed (result_type __s) [inline]`

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

Definition at line 865 of file random.h.

5.433.4.8 `template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r
>::seed () [inline]`

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

Definition at line 854 of file random.h.

5.433.5 Friends And Related Function Documentation

5.433.5.1 `template<typename _RandomNumberEngine, size_t __p,
size_t __r> template<typename _RandomNumberEngine1 ,
size_t __p1, size_t __r1, typename _CharT , typename _Traits
> std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 >
&) [friend]`

Inserts the current state of a discard_block_engine random number generator engine `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A discard_block_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.433.5.2 `template<typename _RandomNumberEngine, size_t __p,
size_t __r> bool operator==(const discard_block_engine<
_RandomNumberEngine, __p, __r > & __lhs, const
discard_block_engine< _RandomNumberEngine, __p, __r > &
__rhs) [friend]`

Compares two discard_block_engine random number generator objects of the same type for equality.

Parameters

`__lhs` A discard_block_engine random number generator object.

`__rhs` Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 940 of file random.h.

5.433.5.3 `template<typename _RandomNumberEngine, size_t __p,
size_t __r> template<typename _RandomNumberEngine1
, size_t __p1, size_t __r1, typename _CharT , typename
_Traits > std::basic_istream<_CharT, _Traits>&
operator>> (std::basic_istream< _CharT, _Traits > & ,
std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 >
&) [friend]`

Extracts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `discard_block_engine` random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.434 `std::discrete_distribution< _IntType >` Class Template Reference

A [discrete_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- `template<typename _InputIterator >
discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)`

- `template<typename _Func >`
discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)
- **discrete_distribution** (const [param_type](#) &__p)
- **discrete_distribution** ([initializer_list](#)< double > __wl)
- [result_type](#) max () const
- [result_type](#) min () const
- `template<typename _UniformRandomNumberGenerator >`
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- `template<typename _UniformRandomNumberGenerator >`
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- `std::vector< double >` [probabilities](#) () const
- void [reset](#) ()

Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >`
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::discrete_distribution](#)< _IntType1 > &)
- `template<typename _IntType1 , typename _CharT , typename _Traits >`
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::discrete_distribution](#)< _IntType1 > &)

5.434.1 Detailed Description

`template<typename _IntType = int> class std::discrete_distribution< _IntType >`

A [discrete_distribution](#) random number distribution. The formula for the discrete probability mass function is

Definition at line 4682 of file random.h.

5.434.2 Member Typedef Documentation

5.434.2.1 `template<typename _IntType = int> typedef _IntType std::discrete_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4689 of file random.h.

5.434.3 Member Function Documentation

5.434.3.1 `template<typename _IntType = int> result_type
std::discrete_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4795 of file random.h.

References std::vector< _Tp, _Alloc >::size().

5.434.3.2 `template<typename _IntType = int> result_type
std::discrete_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4788 of file random.h.

5.434.3.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::discrete_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4803 of file random.h.

References std::discrete_distribution< _IntType >::operator()(), and std::discrete_distribution< _IntType >::param().

Referenced by std::discrete_distribution< _IntType >::operator()().

5.434.3.4 `template<typename _IntType = int> void std::discrete_distribution<
_IntType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 4781 of file random.h.

5.434.3.5 `template<typename _IntType = int> param_type
std::discrete_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4773 of file random.h.

Referenced by `std::discrete_distribution< _IntType >::operator()()`, and `std::operator==()`.

5.434.3.6 `template<typename _IntType = int> std::vector<double>
std::discrete_distribution< _IntType >::probabilities () const
[inline]`

Returns the probabilities of the distribution.

Definition at line 4766 of file random.h.

5.434.3.7 `template<typename _IntType = int> void std::discrete_distribution<
_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 4759 of file random.h.

5.434.4 Friends And Related Function Documentation

5.434.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream< _CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & ,
const std::discrete_distribution< _IntType1 > &) [friend]`

Inserts a `discrete_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `discrete_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.435 `std::discrete_distribution<_IntType>::param_type` Struct Reference 2409

5.434.4.2 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &, std::discrete_distribution<_IntType1> &) [friend]`

Extracts a `discrete_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `discrete_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.435 `std::discrete_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `discrete_distribution<_IntType>` `distribution_type`

Public Member Functions

- `template<typename _InputIterator> param_type (_InputIterator __wbegin, _InputIterator __wend)`
- `template<typename _Func> param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)`
- `param_type (initializer_list<double> __wil)`
- `std::vector<double> probabilities () const`

Friends

- class `discrete_distribution<_IntType>`
- bool `operator==` (const `param_type` &__p1, const `param_type` &__p2)

5.435.1 Detailed Description

template<typename _IntType = int> struct std::discrete_distribution< _IntType >::param_type

Parameter type.

Definition at line 4691 of file random.h.

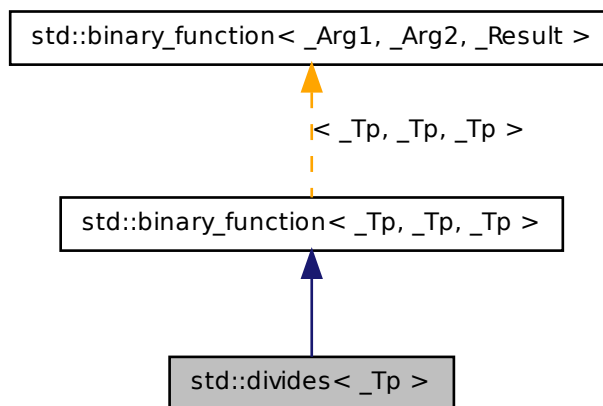
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.436 std::divides< _Tp > Struct Template Reference

One of the [math functions](#).

Inheritance diagram for std::divides< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)

- typedef _Tp [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

5.436.1 Detailed Description

template<typename _Tp> struct std::divides< _Tp >

One of the [math functors](#).

Definition at line 162 of file stl_function.h.

5.436.2 Member Typedef Documentation

5.436.2.1 **typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::first_argument_type [inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.436.2.2 **typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type
[inherited]**

type of the return type

Definition at line 118 of file stl_function.h.

5.436.2.3 **typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::second_argument_type [inherited]**

the type of the second argument

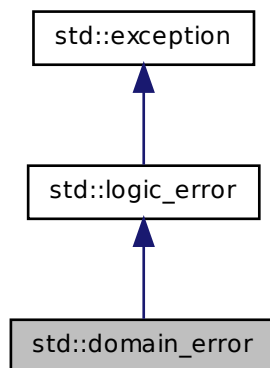
Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.437 std::domain_error Class Reference

Inheritance diagram for std::domain_error:



Public Member Functions

- **domain_error** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.437.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 73 of file stdexcept.

5.437.2 Member Function Documentation

5.437.2.1 virtual const char* std::logic_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

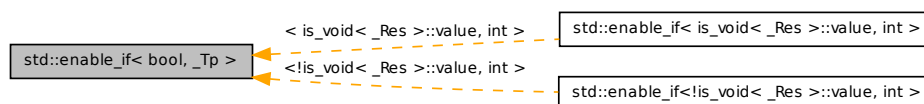
The documentation for this class was generated from the following file:

- [stdexcept](#)

5.438 `std::enable_if< bool, _Tp >` Struct Template Reference

[enable_if](#)

Inheritance diagram for `std::enable_if< bool, _Tp >`:



5.438.1 Detailed Description

```
template<bool, typename _Tp = void> struct std::enable_if< bool, _Tp >
```

[enable_if](#)

Definition at line 384 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.439 `std::enable_shared_from_this< _Tp >` Class Template Reference

Base class allowing use of member function `shared_from_this`.

Public Member Functions

- [shared_ptr](#)< _Tp > **shared_from_this** ()
- [shared_ptr](#)< const _Tp > **shared_from_this** () const

Protected Member Functions

- **enable_shared_from_this** (const [enable_shared_from_this](#) &)
- [enable_shared_from_this](#) & **operator=** (const [enable_shared_from_this](#) &)

Friends

- `template<typename _Tp1 >
void __enable_shared_from_this_helper (const __shared_count<> &__pn,
const enable_shared_from_this *__pe, const _Tp1 *__px)`

5.439.1 Detailed Description

`template<typename _Tp> class std::enable_shared_from_this< _Tp >`

Base class allowing use of member function `shared_from_this`.

Definition at line 441 of file `shared_ptr.h`.

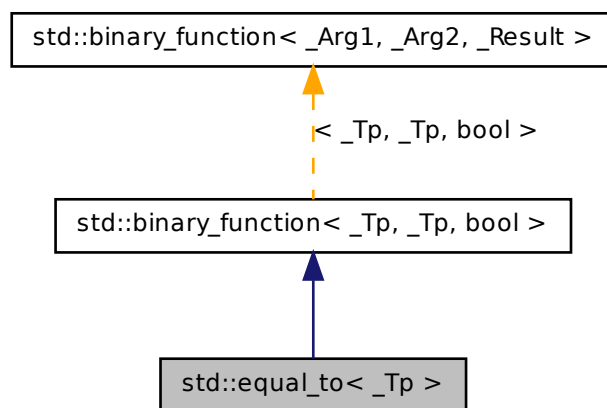
The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

5.440 `std::equal_to< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::equal_to<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.440.1 Detailed Description

`template<typename _Tp> struct std::equal_to<_Tp>`

One of the [comparison functors](#).

Definition at line 199 of file `stl_function.h`.

5.440.2 Member Typedef Documentation

5.440.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.440.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.440.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.441 std::error_category Class Reference

[error_category](#)

Public Member Functions

- **error_category** (const [error_category](#) &)
- virtual [error_condition](#) **default_error_condition** (int __i) const
- virtual bool **equivalent** (const [error_code](#) &__code, int __i) const
- virtual bool **equivalent** (int __i, const [error_condition](#) &__cond) const
- virtual [string](#) **message** (int) const =0
- virtual const char * **name** () const =0
- bool **operator!=** (const [error_category](#) &__other) const
- bool **operator<** (const [error_category](#) &__other) const
- [error_category](#) & **operator=** (const [error_category](#) &)
- bool **operator==** (const [error_category](#) &__other) const

5.441.1 Detailed Description

[error_category](#)

Definition at line 64 of file `system_error`.

The documentation for this class was generated from the following file:

- [system_error](#)

5.442 std::error_code Struct Reference

[error_code](#)

Public Member Functions

- **error_code** (int __v, const [error_category](#) &__cat)
- template<typename _ErrorCodeEnum >
 error_code (_ErrorCodeEnum __e, typename [enable_if](#)< [is_error_code_enum](#)< _ErrorCodeEnum >::value >::type !=0)
- void **assign** (int __v, const [error_category](#) &__cat)
- const [error_category](#) & **category** () const
- void **clear** ()
- [error_condition](#) **default_error_condition** () const
- [string](#) **message** () const
- [operator bool](#) () const
- template<typename _ErrorCodeEnum >
 [enable_if](#)< [is_error_code_enum](#)< _ErrorCodeEnum >::value, [error_code](#) & >::type **operator=** (_ErrorCodeEnum __e)
- int **value** () const

Friends

- class **hash**< [error_code](#) >

5.442.1 Detailed Description

[error_code](#)

Definition at line 116 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system_error](#)

5.443 `std::error_condition` Struct Reference

[error_condition](#)

Public Member Functions

- **error_condition** (int __v, const [error_category](#) &__cat)
- template<typename _ErrorConditionEnum >
error_condition (_ErrorConditionEnum __e, typename [enable_if](#)< [is_error_ - condition_enum](#)< _ErrorConditionEnum >::value >::type !=0)
- void **assign** (int __v, const [error_category](#) &__cat)
- const [error_category](#) & **category** () const
- void **clear** ()
- [string](#) **message** () const
- **operator bool** () const
- template<typename _ErrorConditionEnum >
[enable_if](#)< [is_error_condition_enum](#)< _ErrorConditionEnum >::value, [error_ - condition](#) & >::type **operator=** (_ErrorConditionEnum __e)
- int **value** () const

5.443.1 Detailed Description

[error_condition](#)

Definition at line 193 of file `system_error`.

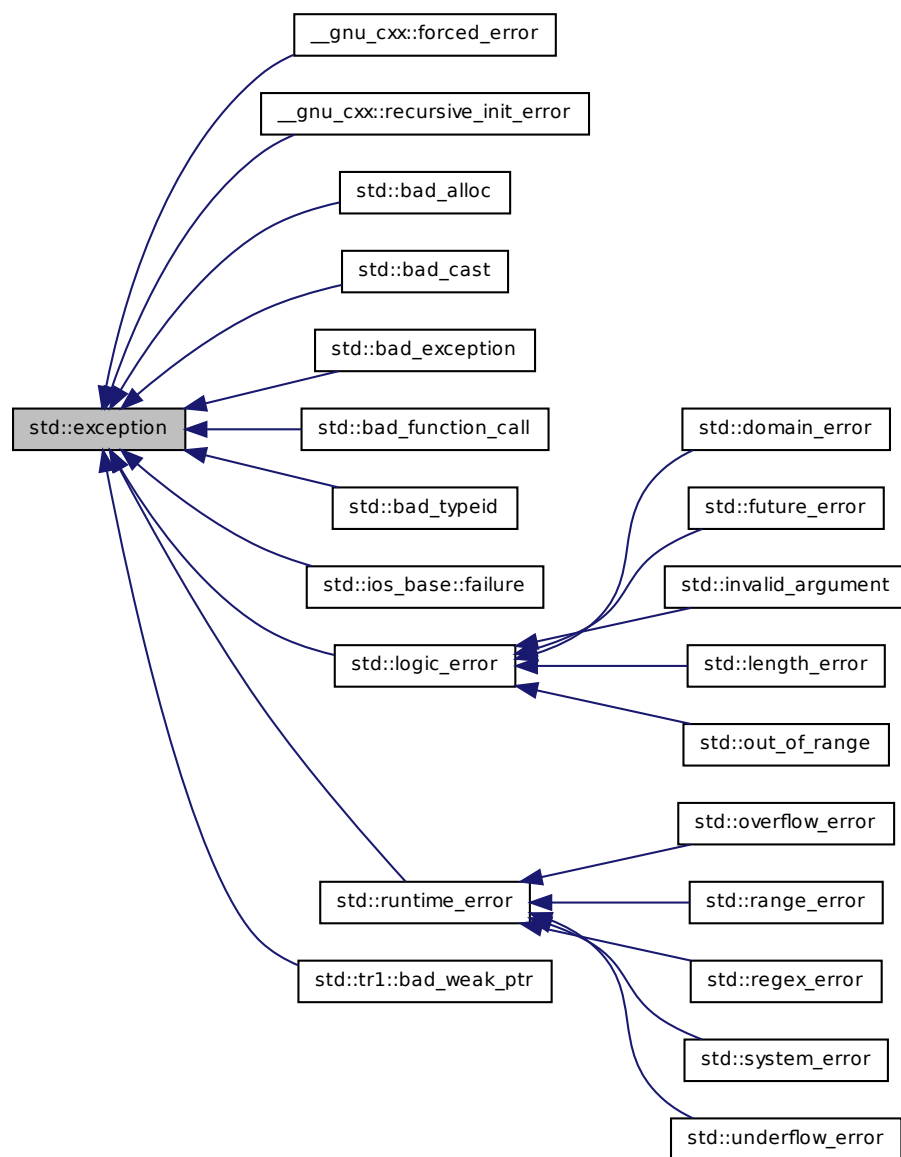
The documentation for this struct was generated from the following file:

- [system_error](#)

5.444 `std::exception` Class Reference

Base class for all library exceptions.

Inheritance diagram for std::exception:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.444.1 Detailed Description

Base class for all library exceptions. This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 61 of file `exception`.

5.444.2 Member Function Documentation

5.444.2.1 virtual const char* std::exception::what () const throw () [[virtual](#)]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and [std::ios_base::failure](#).

The documentation for this class was generated from the following file:

- [exception](#)

5.445 std::exponential_distribution< _RealType > Class Template Reference

An exponential continuous distribution for random numbers.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- `exponential_distribution` (const `result_type` &__lambda=`result_type`(1))
- `exponential_distribution` (const `param_type` &__p)
- `_RealType lambda` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (_UniformRandomNumberGenerator &__urng)
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (_UniformRandomNumberGenerator &__urng, const `param_type` &__p)
- `param_type param` () const
- `void param` (const `param_type` &__param)
- `void reset` ()

5.445.1 Detailed Description

`template<typename _RealType = double> class std::exponential_distribution<_RealType>`

An exponential continuous distribution for random numbers. The formula for the exponential probability density function is $p(x|\lambda) = \lambda e^{-\lambda x}$.

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	<i>zero</i>
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda^2}$

Table 5.1: Distribution Statistics

Definition at line 4161 of file random.h.

5.445.2 Member Typedef Documentation

5.445.2.1 `template<typename _RealType = double> typedef _RealType std::exponential_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4168 of file random.h.

5.445.3 Constructor & Destructor Documentation

5.445.3.1 `template<typename _RealType = double> std::exponential_distribution< _RealType >::exponential_distribution (const result_type & __lambda = result_type (1)) [inline, explicit]`

Constructs an exponential distribution with inverse scale parameter λ .

Definition at line 4199 of file random.h.

5.445.4 Member Function Documentation

5.445.4.1 `template<typename _RealType = double> _RealType std::exponential_distribution< _RealType >::lambda () const [inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4220 of file random.h.

5.445.4.2 `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4249 of file random.h.

5.445.4.3 `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4242 of file random.h.

5.445.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::exponential_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4257 of file random.h.

5.445 `std::exponential_distribution<_RealType>` Class Template Reference 2423

References `std::exponential_distribution<_RealType>::operator()()`, and `std::exponential_distribution<_RealType>::param()`.

Referenced by `std::exponential_distribution<_RealType>::operator()()`.

5.445.4.5 `template<typename _RealType = double> param_type
std::exponential_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4227 of file `random.h`.

Referenced by `std::exponential_distribution<_RealType>::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.445.4.6 `template<typename _RealType = double> void
std::exponential_distribution<_RealType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4235 of file `random.h`.

5.445.4.7 `template<typename _RealType = double> void
std::exponential_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4214 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

5.446 `std::exponential_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [exponential_distribution< _RealType >](#) **distribution_type**

Public Member Functions

- param_type** ([_RealType](#) __lambda=[_RealType](#)(1))
- [_RealType](#) **lambda** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.446.1 Detailed Description

`template<typename _RealType = double> struct std::exponential_distribution< _RealType >::param_type`

Parameter type.

Definition at line 4170 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.447 `std::extreme_value_distribution< _RealType >` Class Template Reference

A [extreme_value_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **extreme_value_distribution** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **extreme_value_distribution** (const [param_type](#) &__p)
- _RealType **a** () const
- _RealType **b** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- void **param** (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void **reset** ()

5.447.1 Detailed Description

template<typename _RealType = double> class std::extreme_value_distribution<_RealType>

A [extreme_value_distribution](#) random number distribution. The formula for the normal probability mass function is

$$p(x|a, b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 4510 of file random.h.

5.447.2 Member Typedef Documentation

5.447.2.1 **template<typename _RealType = double> typedef _RealType std::extreme_value_distribution<_RealType>::result_type**

The type of the range of the distribution.

Definition at line 4517 of file random.h.

5.447.3 Member Function Documentation

5.447.3.1 `template<typename _RealType = double> _RealType
std::extreme_value_distribution< _RealType >::a () const
[inline]`

Return the a parameter of the distribution.

Definition at line 4568 of file random.h.

5.447.3.2 `template<typename _RealType = double> _RealType
std::extreme_value_distribution< _RealType >::b () const
[inline]`

Return the b parameter of the distribution.

Definition at line 4575 of file random.h.

5.447.3.3 `template<typename _RealType = double> result_type
std::extreme_value_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4604 of file random.h.

5.447.3.4 `template<typename _RealType = double> result_type
std::extreme_value_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4597 of file random.h.

5.447.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::extreme_value_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4612 of file random.h.

References `std::extreme_value_distribution< _RealType >::operator()()`, and `std::extreme_value_distribution< _RealType >::param()`.

5.448 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference 2427

Referenced by `std::extreme_value_distribution<_RealType>::operator()`.

5.447.3.6 `template<typename _RealType = double> void
std::extreme_value_distribution<_RealType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4590 of file `random.h`.

5.447.3.7 `template<typename _RealType = double> param_type
std::extreme_value_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4582 of file `random.h`.

Referenced by `std::extreme_value_distribution<_RealType>::operator()`, `std::operator==()`, and `std::operator>>()`.

5.447.3.8 `template<typename _RealType = double> void
std::extreme_value_distribution<_RealType>::reset ()
[inline]`

Resets the distribution state.

Definition at line 4561 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.448 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `extreme_value_distribution<_RealType>` `distribution_type`

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.448.1 Detailed Description

template<typename _RealType = double> struct std::extreme_value_distribution< _RealType >::param_type

Parameter type.

Definition at line 4519 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.449 std::fisher_f_distribution< _RealType > Class Template Reference

A [fisher_f_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **fisher_f_distribution** (_RealType __m=_RealType(1), _RealType __n=_RealType(1))
- **fisher_f_distribution** (const [param_type](#) &__p)

- `_RealType m () const`
- `result_type max () const`
- `result_type min () const`
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng, const
param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &, const std::fisher_f_distribution<_RealType1> &)`
- `template<typename _RealType1 >
bool operator== (const std::fisher_f_distribution<_RealType1> &__d1, const
std::fisher_f_distribution<_RealType1> &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::fisher_f_distribution<_RealType1> &)`

5.449.1 Detailed Description

`template<typename _RealType = double> class std::fisher_f_distribution<_RealType>`

A `fisher_f_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 2877 of file `random.h`.

5.449.2 Member Typedef Documentation

5.449.2.1 `template<typename _RealType = double> typedef _RealType
std::fisher_f_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2884 of file random.h.

5.449.3 Member Function Documentation

5.449.3.1 `template<typename _RealType = double> result_type
std::fisher_f_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2971 of file random.h.

5.449.3.2 `template<typename _RealType = double> result_type
std::fisher_f_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2964 of file random.h.

5.449.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::fisher_f_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2979 of file random.h.

5.449.3.4 `template<typename _RealType = double> void
std::fisher_f_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2957 of file random.h.

5.449.3.5 `template<typename _RealType = double> param_type
std::fisher_f_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2949 of file random.h.

5.449.3.6 `template<typename _RealType = double> void
std::fisher_f_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2928 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

5.449.4 Friends And Related Function Documentation

5.449.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::fisher_f_distribution< _RealType1 > &) [friend]`

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `fisher_f_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.449.4.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::fisher_f_distribution<
_RealType1 > & __d1, const std::fisher_f_distribution< _RealType1
> & __d2) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3000 of file random.h.

5.449.4.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::fisher_f_distribution< _RealType1 > &) [friend]`

Extracts a fisher_f_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A fisher_f_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.450 `std::fisher_f_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [fisher_f_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(1), _RealType __n=_RealType(1))
- _RealType **m** () const
- _RealType **n** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.450.1 Detailed Description

```
template<typename _RealType = double> struct std::fisher_f_distribution< _-  
_RealType >::param_type
```

Parameter type.

Definition at line 2886 of file random.h.

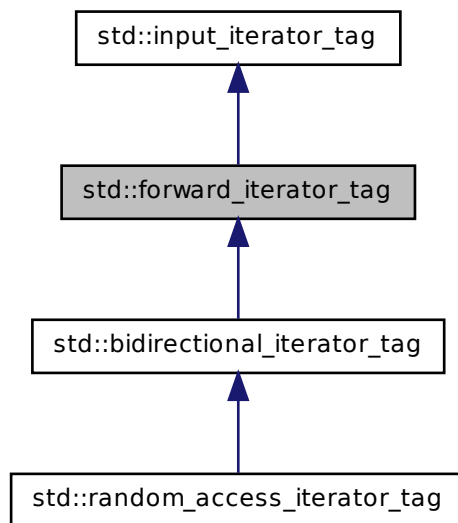
The documentation for this struct was generated from the following file:

- [random.h](#)

5.451 std::forward_iterator_tag Struct Reference

Forward iterators support a superset of input iterator operations.

Inheritance diagram for std::forward_iterator_tag:



5.451.1 Detailed Description

Forward iterators support a superset of input iterator operations.

Definition at line 84 of file `stl_iterator_base_types.h`.

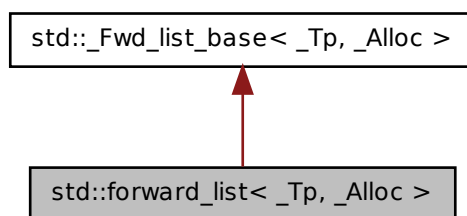
The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.452 `std::forward_list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::forward_list< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Fwd_list_const_iterator< _Tp >` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::ptrdiff_t` **difference_type**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**

- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [forward_list](#) (const _Alloc &__al=_Alloc())
- [forward_list](#) (const [forward_list](#) &__list, const _Alloc &__al)
- [forward_list](#) (size_type __n)
- [forward_list](#) ([forward_list](#) &&__list)
- [forward_list](#) (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- [forward_list](#) (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- [forward_list](#) ([forward_list](#) &&__list, const _Alloc &__al)
- template<typename _InputIterator >
[forward_list](#) (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- [forward_list](#) (const [forward_list](#) &__list)
- [~forward_list](#) ()
- void [assign](#) (std::initializer_list< _Tp > __il)
- template<typename _InputIterator >
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (size_type __n, const _Tp &__val)
- [iterator before_begin](#) ()
- [const_iterator before_begin](#) () const
- [iterator begin](#) ()
- [const_iterator begin](#) () const
- [const_iterator cbefore_begin](#) () const
- [const_iterator cbegin](#) () const
- [const_iterator cend](#) () const
- void [clear](#) ()
- template<typename... _Args>
[iterator emplace_after](#) (const_iterator __pos, _Args &&...__args)
- template<typename... _Args>
void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const
- [const_iterator end](#) () const
- [iterator end](#) ()
- void [erase_after](#) (const_iterator __pos)
- void [erase_after](#) (const_iterator __pos, const_iterator __last)
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const

- `template<typename _InputIterator >`
`iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator`
`__last)`
- `iterator insert_after (const_iterator __pos, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, size_type __n, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, std::initializer_list< _Tp > __il)`
- `size_type max_size () const`
- `template<typename _Comp >`
`void merge (forward_list &&__list, _Comp __comp)`
- `void merge (forward_list &&__list)`
- `forward_list & operator= (std::initializer_list< _Tp > __il)`
- `forward_list & operator= (const forward_list &&__list)`
- `forward_list & operator= (forward_list &&__list)`
- `void pop_front ()`
- `void push_front (const _Tp &&__val)`
- `void push_front (_Tp &&__val)`
- `void remove (const _Tp &&__val)`
- `template<typename _Pred >`
`void remove_if (_Pred __pred)`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, value_type __val)`
- `void reverse ()`
- `template<typename _Comp >`
`void sort (_Comp __comp)`
- `void sort ()`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __-`
`__before, const_iterator __last)`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __-`
`__i)`
- `void splice_after (const_iterator __pos, forward_list &&__list)`
- `void swap (forward_list &&__list)`
- `template<typename _BinPred >`
`void unique (_BinPred __binary_pred)`
- `void unique ()`

Private Types

- `typedef _Alloc::template rebind< _Fwd_list_node< _Tp > >::other _Node_-`
`alloc_type`

Private Member Functions

- `template<typename... _Args>
_Node * _M_create_node (_Args &&...__args)`
- `void _M_erase_after (_Fwd_list_node_base *__pos)`
- `void _M_erase_after (_Fwd_list_node_base *__pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `const _Node_alloc_type & _M_get_Node_allocator () const`
- `_Node_alloc_type & _M_get_Node_allocator ()`
- `template<typename... _Args>
_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void _M_put_node (_Node * __p)`

Private Attributes

- `_Fwd_list_impl _M_impl`

5.452.1 Detailed Description

`template<typename _Tp, typename _Alloc = allocator<_Tp>> class std::forward_list<_Tp, _Alloc>`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `forward_list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `Fwd_list_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `forward_list<X,Alloc1>` are spliced into `forward_list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Definition at line 407 of file `forward_list.h`.

5.452.2 Constructor & Destructor Documentation

5.452.2.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (const _Alloc & __al
= _Alloc()) [inline, explicit]`

Creates a forward_list with no elements.

Parameters

al An allocator object.

Definition at line 436 of file forward_list.h.

5.452.2.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (const forward_list<
_Tp, _Alloc> & __list, const _Alloc & __al) [inline]`

Copy constructor with allocator argument.

Parameters

list Input list to copy.

al An allocator object.

Definition at line 445 of file forward_list.h.

5.452.2.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (forward_list<_Tp,
_Alloc> && __list, const _Alloc & __al) [inline]`

Move constructor with allocator argument.

Parameters

list Input list to move.

al An allocator object.

Definition at line 454 of file forward_list.h.

5.452.2.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (size_type __n)
[inline, explicit]`

Creates a forward_list with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor creates the forward_list with *n* default constructed elements.

Definition at line 466 of file forward_list.h.

```
5.452.2.5  template<typename _Tp, typename _Alloc = allocator<_Tp>>
            std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n,
            const _Tp & __value, const _Alloc & __al = _Alloc() )
            [inline]
```

Creates a forward_list with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

al An allocator object.

This constructor fills the forward_list with *n* copies of *value*.

Definition at line 479 of file forward_list.h.

```
5.452.2.6  template<typename _Tp, typename _Alloc = allocator<_Tp>>
            template<typename _InputIterator > std::forward_list< _Tp, _Alloc
            >::forward_list ( _InputIterator __first, _InputIterator __last,
            const _Alloc & __al = _Alloc() ) [inline]
```

Builds a forward_list from a range.

Parameters

first An input iterator.

last An input iterator.

al An allocator object.

Create a forward_list consisting of copies of the elements from [*first*,*last*). This is linear in N (where N is distance(*first*,*last*)).

Definition at line 495 of file forward_list.h.

5.452.2.7 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (const forward_list<
_Tp, _Alloc> & __list) [inline]`

The forward_list copy constructor.

Parameters

list A forward_list of identical element and allocator types.

The newly-created forward_list uses a copy of the allocation object used by *list*.

Definition at line 512 of file forward_list.h.

References std::forward_list<_Tp, _Alloc>::begin(), and std::forward_list<_Tp, _Alloc>::end().

5.452.2.8 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (forward_list<_Tp,
_Alloc> && __list) [inline]`

The forward_list move constructor.

Parameters

list A forward_list of identical element and allocator types.

The newly-created forward_list contains the exact contents of *forward_list*. The contents of *list* are a valid, but unspecified forward_list.

Definition at line 525 of file forward_list.h.

5.452.2.9 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (std::initializer_list<
_Tp> __il, const _Alloc & __al = _Alloc()) [inline]`

Builds a forward_list from an [initializer_list](#).

Parameters

il An [initializer_list](#) of value_type.

al An allocator object.

Create a forward_list consisting of copies of the elements in the [initializer_list](#) *il*. This is linear in il.size().

Definition at line 536 of file forward_list.h.

5.452.2.10 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~~forward_list () [inline]`

The `forward_list` dtor.

Definition at line 544 of file `forward_list.h`.

5.452.3 Member Function Documentation

5.452.3.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator > void std::forward_list< _Tp,
_Alloc >::assign (_InputIterator __first, _InputIterator __last)
[inline]`

Assigns a range to a `forward_list`.

Parameters

first An input iterator.

last An input iterator.

This function fills a `forward_list` with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 606 of file `forward_list.h`.

5.452.3.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::assign (size_type __n, const _Tp
& __val) [inline]`

Assigns a given value to a `forward_list`.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a `forward_list` with *n* copies of the given value. Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 623 of file `forward_list.h`.

5.452.3.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list<_Tp, _Alloc>::assign (std::initializer_list<_Tp
> __il) [inline]`

Assigns an [initializer_list](#) to a forward_list.

Parameters

il An [initializer_list](#) of value_type.

Replace the contents of the forward_list with copies of the elements in the [initializer_list](#) *il*. This is linear in *il.size()*.

Definition at line 638 of file forward_list.h.

5.452.3.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::before_begin ()
[inline]`

Returns a read/write iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 656 of file forward_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::forward_list<_Tp, _Alloc>::resize()`.

5.452.3.5 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::before_begin ()
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 665 of file forward_list.h.

5.452.3.6 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::begin () [inline]`

Returns a read/write iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 673 of file forward_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`, `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::forward_list<_Tp, _Alloc>::unique()`.

5.452.3.7 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::begin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 682 of file forward_list.h.

5.452.3.8 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin ()
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 718 of file forward_list.h.

5.452.3.9 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbegin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 709 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::operator==().

5.452.3.10 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cend () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 727 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::operator==().

5.452.3.11 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::clear () [inline]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves

are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1016 of file forward_list.h.

```
5.452.3.12  template<typename _Tp, typename _Alloc = allocator<_Tp>>
               template<typename... _Args> iterator std::forward_list<_Tp,
               _Alloc>::emplace_after ( const_iterator __pos, _Args &&...
               __args ) [inline]
```

Constructs object in forward_list after the specified iterator.

Parameters

pos A const_iterator into the forward_list.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 840 of file forward_list.h.

```
5.452.3.13  template<typename _Tp, typename _Alloc = allocator<_Tp>>
               template<typename... _Args> void std::forward_list<_Tp, _Alloc
               >::emplace_front ( _Args &&... __args ) [inline]
```

Constructs object in forward_list at the front of the list.

Parameters

args Arguments.

This function will insert an object of type Tp constructed with Tp(std::forward<Args>(args)...) at the front of the list. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 784 of file forward_list.h.

5.452.3.14 `template<typename _Tp, typename _Alloc = allocator<_Tp>> bool
std::forward_list< _Tp, _Alloc >::empty () const [inline]`

Returns true if the forward_list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 735 of file forward_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

5.452.3.15 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 691 of file forward_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::resize()`, and `std::forward_list< _Tp, _Alloc >::unique()`.

5.452.3.16 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::end () const
[inline]`

Returns a read-only iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 700 of file forward_list.h.

5.452.3.17 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::erase_after (const_iterator
__pos, const_iterator __last) [inline]`

Remove a range of elements.

Parameters

pos Iterator pointing before the first element to be erased.

last Iterator pointing to one past the last element to be erased.

This function will erase the elements in the range (pos,last) and shorten the forward_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only

erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 960 of file forward_list.h.

5.452.3.18 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list<_Tp, _Alloc>::erase_after (const_iterator
__pos) [inline]`

Removes the element pointed to by the iterator following `pos`.

Parameters

pos Iterator pointing before element to be erased.

This function will erase the element at the given position and thus shorten the forward_list by one.

Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 939 of file forward_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, `std::forward_list<_Tp, _Alloc>::resize()`, and `std::forward_list<_Tp, _Alloc>::unique()`.

5.452.3.19 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list<_Tp, _Alloc>::front () [inline]`

Returns a read/write reference to the data at the first element of the forward_list.

Definition at line 752 of file forward_list.h.

5.452.3.20 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list<_Tp, _Alloc>::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the forward_list.

Definition at line 763 of file forward_list.h.

5.452.3.21 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ()
const [inline]`

Get a copy of the memory allocation object.

Definition at line 646 of file forward_list.h.

5.452.3.22 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
const_iterator __pos, const _Tp & __val) [inline]`

Inserts given value into forward_list after specified iterator.

Parameters

pos An iterator into the forward_list.

val Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 857 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::forward_list< _Tp, _Alloc >::resize().

5.452.3.23 `template<typename _Tp, typename _Alloc > forward_list< _Tp,
_Alloc >::iterator std::forward_list< _Tp, _Alloc >::insert_after (
const_iterator __pos, size_type __n, const _Tp & __val)`

Inserts a number of copies of given data into the forward_list.

Parameters

pos An iterator into the forward_list.

n Number of elements to be inserted.

val Data to be inserted.

Returns

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 244 of file forward_list.tcc.

5.452.3.24 `template<typename _Tp, typename _Alloc > template<typename
_InputIterator > forward_list< _Tp, _Alloc >::iterator
std::forward_list< _Tp, _Alloc >::insert_after (const_iterator
__pos, _InputIterator __first, _InputIterator __last)`

Inserts a range into the forward_list.

Parameters

position An iterator into the forward_list.

first An input iterator.

last An input iterator.

Returns

An iterator pointing to the last inserted element or *pos* if *first* == *last*.

This function will insert copies of the data in the range [*first*,*last*) into the forward_list after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 259 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::empty().

5.452.3.25 `template<typename _Tp, typename _Alloc > forward_list< _Tp,
_Alloc >::iterator std::forward_list< _Tp, _Alloc >::insert_after (
const_iterator __pos, std::initializer_list< _Tp > __il)`

Inserts the contents of an [initializer_list](#) into forward_list after the specified iterator.

Parameters

pos An iterator into the forward_list.

il An [initializer_list](#) of value_type.

Returns

An iterator pointing to the last inserted element or *pos* if *il* is empty.

This function will insert copies of the data in the [initializer_list](#) *il* into the `forward_list` before the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 272 of file `forward_list.tcc`.

5.452.3.26 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size () const
[inline]`

Returns the largest possible size of `forward_list`.

Definition at line 742 of file `forward_list.h`.

5.452.3.27 `template<typename _Tp , typename _Alloc > template<typename
_Comp > void std::forward_list< _Tp, _Alloc >::merge (
forward_list< _Tp, _Alloc > && __list, _Comp __comp)`

Merge sorted lists according to comparison function.

Parameters

list Sorted list to merge.

comp Comparison function defining sort order.

Assumes that both *list* and this list are sorted according to *comp*. Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equivalent according to *comp*().

Definition at line 339 of file `forward_list.tcc`.

5.452.3.28 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::merge (forward_list< _Tp,
_Alloc > && __list) [inline]`

Merge sorted lists.

Parameters

list Sorted list to merge.

Assumes that both *list* and this list are sorted according to operator<(). Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equal.

Definition at line 1147 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::merge().

Referenced by std::forward_list< _Tp, _Alloc >::merge().

5.452.3.29 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list< _Tp, _Alloc >::operator= (
forward_list< _Tp, _Alloc > && __list) [inline]`

The forward_list move assignment operator.

Parameters

list A forward_list of identical element and allocator types.

The contents of *list* are moved into this forward_list (without copying). *list* is a valid, but unspecified forward_list

Definition at line 568 of file forward_list.h.

5.452.3.30 `template<typename _Tp, typename _Alloc > forward_list< _Tp,
_Alloc > & std::forward_list< _Tp, _Alloc >::operator= (const
forward_list< _Tp, _Alloc > & __list)`

The forward_list assignment operator.

Parameters

list A forward_list of identical element and allocator types.

All the elements of *list* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 140 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::before_begin(), std::forward_list< _Tp, _Alloc >::begin(), std::forward_list< _Tp, _Alloc >::cbegin(), std::forward_list< _Tp, _Alloc >::cend(), std::forward_list< _Tp, _Alloc >::end(), std::forward_list< _Tp, _Alloc >::erase_after(), and std::forward_list< _Tp, _Alloc >::insert_after().

5.452.3.31 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list< _Tp, _Alloc >::operator= (
std::initializer_list< _Tp > __il) [inline]`

The forward_list initializer list assignment operator.

Parameters

il An [initializer_list](#) of value_type.

Replace the contents of the forward_list with copies of the elements in the [initializer_list](#) *il*. This is linear in *il.size()*.

Definition at line 586 of file forward_list.h.

5.452.3.32 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::pop_front () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the forward_list by one. Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop_front\(\)](#) is called.

Definition at line 822 of file forward_list.h.

5.452.3.33 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::push_front (const _Tp & __val
) [inline]`

Add data to the front of the forward_list.

Parameters

val Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward_list and assigns the given data to it. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 799 of file forward_list.h.

5.452.3.34 `template<typename _Tp, typename _Alloc > void
std::forward_list< _Tp, _Alloc >::remove (const _Tp & __val)`

Remove all elements equal to value.

Parameters

val The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 286 of file forward_list.tcc.

5.452.3.35 `template<typename _Tp, typename _Alloc > template<typename
_Pred > void std::forward_list< _Tp, _Alloc >::remove_if (_Pred
__pred)`

Remove all elements satisfying a predicate.

Parameters

pred Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 302 of file forward_list.tcc.

5.452.3.36 `template<typename _Tp, typename _Alloc > void
std::forward_list< _Tp, _Alloc >::resize (size_type __sz)`

Resizes the forward_list to the specified number of elements.

Parameters

sz Number of elements the forward_list should contain.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current size the forward_list is truncated, otherwise the forward_list is extended and the new elements are default constructed.

Definition at line 185 of file forward_list.tcc.

References `std::forward_list< _Tp, _Alloc >::before_begin()`, `std::forward_list< _Tp, _Alloc >::end()`, and `std::forward_list< _Tp, _Alloc >::erase_after()`.

5.452.3.37 `template<typename _Tp, typename _Alloc > void
std::forward_list< _Tp, _Alloc >::resize (size_type __sz,
value_type __val)`

Resizes the forward_list to the specified number of elements.

Parameters

sz Number of elements the forward_list should contain.

val Data with which new elements should be populated.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current size the forward_list is truncated, otherwise the forward_list is extended and new elements are populated with given data.

Definition at line 204 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::before_begin(), std::forward_list< _Tp, _Alloc >::end(), std::forward_list< _Tp, _Alloc >::erase_after(), and std::forward_list< _Tp, _Alloc >::insert_after().

5.452.3.38 **template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::reverse () [inline]**

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1191 of file forward_list.h.

5.452.3.39 **template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::sort () [inline]**

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1172 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::sort().

Referenced by std::forward_list< _Tp, _Alloc >::sort().

5.452.3.40 **template<typename _Tp, class _Alloc > template<typename _Comp > void std::forward_list< _Tp, _Alloc >::sort (_Comp __comp)**

Sort the [forward_list](#) using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 385 of file forward_list.tcc.

5.452.3.41 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator
__pos, forward_list< _Tp, _Alloc > && __list) [inline]`

Insert contents of another forward_list.

Parameters

pos Iterator referencing the element to insert after.

list Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*.
list becomes an empty list.

Requires this != *x*.

Definition at line 1033 of file forward_list.h.

5.452.3.42 `template<typename _Tp , typename _Alloc > void
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator
__pos, forward_list< _Tp, _Alloc > && __list, const_iterator
__before, const_iterator __last)`

Insert range from another forward_list.

Parameters

pos Iterator referencing the element to insert after.

list Source list.

before Iterator referencing before the start of range in list.

last Iterator referencing the end of range in list.

Removes elements in the range (before,last) and inserts them after *pos* in constant time.

Undefined if *pos* is in (before,last).

Definition at line 233 of file forward_list.tcc.

5.452.3.43 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator
__pos, forward_list< _Tp, _Alloc > && __list, const_iterator __i
) [inline]`

Insert element from another forward_list.

Parameters

pos Iterator referencing the element to insert after.

list Source list.

i Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 1050 of file forward_list.h.

5.452.3.44 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (forward_list< _Tp,
_Alloc > & __list) [inline]`

Swaps data with another forward_list.

Parameters

list A forward_list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 975 of file forward_list.h.

Referenced by `std::swap()`.

5.452.3.45 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::unique () [inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1118 of file forward_list.h.

References `std::forward_list< _Tp, _Alloc >::unique()`.

Referenced by `std::forward_list< _Tp, _Alloc >::unique()`.

5.452.3.46 `template<typename _Tp , typename _Alloc > template<typename
_BinPred > void std::forward_list< _Tp, _Alloc >::unique (
_BinPred __binary_pred)`

Remove consecutive elements satisfying a predicate.

Parameters

binary_pred Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 318 of file forward_list.tcc.

References `std::forward_list< _Tp, _Alloc >::begin()`, `std::forward_list< _Tp, _Alloc >::end()`, and `std::forward_list< _Tp, _Alloc >::erase_after()`.

The documentation for this class was generated from the following files:

- [forward_list.h](#)
- [forward_list.tcc](#)

5.453 std::fpos< _StateT > Class Template Reference

Class representing stream positions.

Public Member Functions

- [fpos](#) (streamoff __off)
- [operator streamoff](#) () const
- [fpos operator+](#) (streamoff __off) const
- [fpos & operator+=](#) (streamoff __off)
- [fpos operator-](#) (streamoff __off) const
- [streamoff operator-](#) (const [fpos](#) &__other) const
- [fpos & operator-=](#) (streamoff __off)
- void [state](#) (_StateT __st)
- _StateT [state](#) () const

5.453.1 Detailed Description

`template<typename _StateT> class std::fpos< _StateT >`

Class representing stream positions. The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

Parameters

StateT Type passed to and returned from [state\(\)](#).

Definition at line 112 of file postypes.h.

5.453.2 Constructor & Destructor Documentation

5.453.2.1 `template<typename _StateT> std::fpos< _StateT >::fpos (streamoff __off) [inline]`

Construct position from offset.

Definition at line 133 of file postypes.h.

5.453.3 Member Function Documentation

5.453.3.1 `template<typename _StateT> std::fpos< _StateT >::operator streamoff () const [inline]`

Convert to streamoff.

Definition at line 137 of file postypes.h.

5.453.3.2 `template<typename _StateT> fpos std::fpos< _StateT >::operator+ (streamoff __off) const [inline]`

Add position and offset.

Definition at line 178 of file postypes.h.

5.453.3.3 `template<typename _StateT> fpos& std::fpos< _StateT >::operator+=(streamoff __off) [inline]`

Add offset to this position.

Definition at line 154 of file postypes.h.

5.453.3.4 `template<typename _StateT> streamoff std::fpos< _StateT >::operator- (const fpos< _StateT > & __other) const [inline]`

Subtract position to return offset.

Definition at line 205 of file postypes.h.

5.453.3.5 `template<typename _StateT> fpos std::fpos< _StateT >::operator- (streamoff __off) const [inline]`

Subtract offset from position.

Definition at line 192 of file postypes.h.

5.453.3.6 `template<typename _StateT> fpos& std::fpos< _StateT >::operator-= (streamoff __off) [inline]`

Subtract offset from this position.

Definition at line 165 of file postypes.h.

5.453.3.7 `template<typename _StateT> _StateT std::fpos< _StateT >::state () const [inline]`

Return the last set value of *st*.

Definition at line 146 of file postypes.h.

5.453.3.8 `template<typename _StateT> void std::fpos< _StateT >::state (_StateT __st) [inline]`

Remember the value of *st*.

Definition at line 141 of file postypes.h.

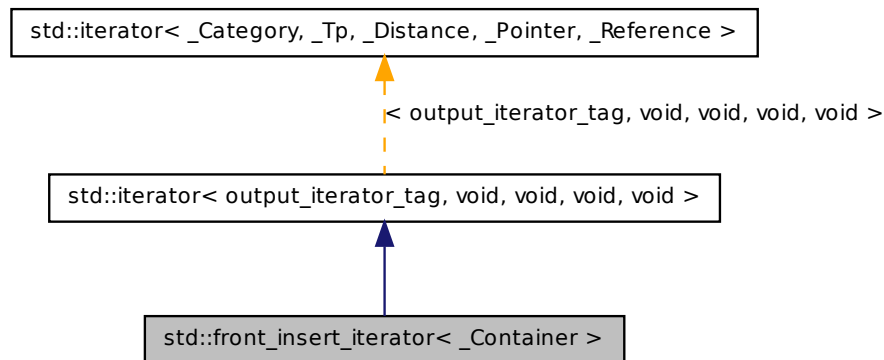
The documentation for this class was generated from the following file:

- [postypes.h](#)

5.454 `std::front_insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::front_insert_iterator<_Container>`:



Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- `front_insert_iterator` (`_Container &__x`)
- `front_insert_iterator` & `operator*` ()
- `front_insert_iterator` & `operator++` ()
- `front_insert_iterator` `operator++` (int)
- `front_insert_iterator` & `operator=` (typename `_Container::const_reference` __-value)
- `front_insert_iterator` & `operator=` (typename `_Container::value_type` &&__-value)

Protected Attributes

- `_Container * container`

5.454.1 Detailed Description

template<typename _Container> class std::front_insert_iterator<_Container >

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using push_front.

Tip: Using the front_inserter function to create these iterators can save typing.

Definition at line 478 of file stl_iterator.h.

5.454.2 Member Typedef Documentation

**5.454.2.1 template<typename _Container > typedef _Container
std::front_insert_iterator<_Container >::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 486 of file stl_iterator.h.

**5.454.2.2 typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::difference_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

**5.454.2.3 typedef output_iterator_tag std::iterator< output_iterator_tag , void
, void , void , void >::iterator_category [inherited]**

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

**5.454.2.4 typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::pointer [inherited]**

This type represents a pointer-to-value_type.

Definition at line 116 of file stl_iterator_base_types.h.

5.454 std::front_insert_iterator< _Container > Class Template Reference 2461

5.454.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.454.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file stl_iterator_base_types.h.

5.454.3 Constructor & Destructor Documentation

5.454.3.1 `template<typename _Container > std::front_insert_iterator< _Container >::front_insert_iterator (_Container & __x) [inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 489 of file stl_iterator.h.

5.454.4 Member Function Documentation

5.454.4.1 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator* () [inline]`

Simply returns *this.

Definition at line 520 of file stl_iterator.h.

5.454.4.2 `template<typename _Container > front_insert_iterator std::front_insert_iterator< _Container >::operator++ (int) [inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 530 of file stl_iterator.h.

5.454.4.3 `template<typename _Container > front_insert_iterator&
std::front_insert_iterator< _Container >::operator++ ()
[inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 525 of file stl_iterator.h.

5.454.4.4 `template<typename _Container > front_insert_iterator&
std::front_insert_iterator< _Container >::operator= (typename
_Container::const_reference __value) [inline]`

Parameters

value An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

Definition at line 503 of file stl_iterator.h.

The documentation for this class was generated from the following file:

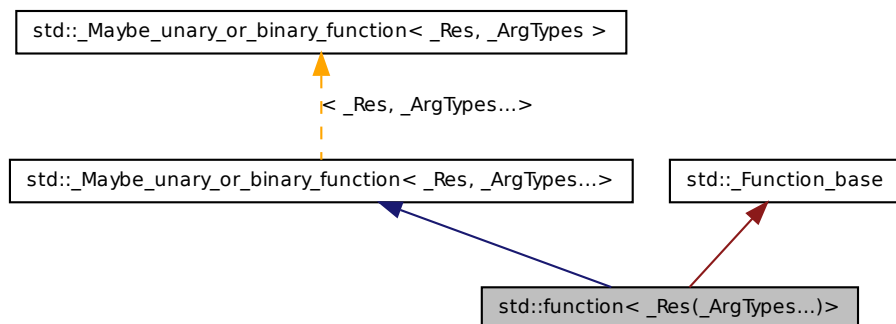
- [stl_iterator.h](#)

5.455 `std::function< _Res(_ArgTypes...)>` Class Template Reference

Primary class template for std::function.

Polymorphic function wrapper.

Inheritance diagram for std::function< _Res(_ArgTypes...)>:



Public Types

- typedef `_Res` **result_type**

Public Member Functions

- `function` ()
- `function` (nullptr_t)
- `function` (function &&__x)
- template<typename _Functor >
`function` (_Functor __f, typename `enable_if`< !is_integral< _Functor >::value, _Useless >::type=_Useless())
- `function` (const function &__x)
- `operator bool` () const
- template<typename _Res2 , typename... _ArgTypes2>
void **operator!=** (const function< _Res2(_ArgTypes2...)> &) const
- `_Res` `operator()` (_ArgTypes...__args) const
- function & `operator=` (nullptr_t)
- template<typename _Functor >
`enable_if`<!is_integral< _Functor >::value, function & >::type `operator=` (_Functor &&__f)
- template<typename _Functor >
`enable_if`<!is_integral< _Functor >::value, function & >::type `operator=` (reference_wrapper< _Functor > __f)

- function & [operator=](#) (const function &__x)
- function & [operator=](#) (function &&__x)
- template<typename _Res2 , typename... _ArgTypes2>
void **operator==** (const function< _Res2(_ArgTypes2...)> &) const
- void [swap](#) (function &__x)
- template<typename _Functor >
_Functor * [target](#) ()
- template<typename _Functor >
const _Functor * [target](#) () const
- const [type_info](#) & [target_type](#) () const

Private Types

- typedef bool(* **_Manager_type**)(_Any_data &, const _Any_data &, _Manager_operation)

Private Member Functions

- bool **_M_empty** () const

Private Attributes

- _Any_data **_M_functor**
- _Manager_type **_M_manager**

Static Private Attributes

- static const std::size_t **_M_max_align**
- static const std::size_t **_M_max_size**

5.455.1 Detailed Description

template<typename _Res, typename... _ArgTypes> class std::function< _Res(_ArgTypes...)>

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 1772 of file functional.

5.455.2 Constructor & Destructor Documentation

5.455.2.1 `template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::function () [inline, explicit]`

Default construct creates an empty function call wrapper.

Postcondition

`!(bool)*this`

Definition at line 1790 of file functional.

5.455.2.2 `template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::function (nullptr_t) [inline]`

Creates an empty function call wrapper.

Postcondition

`!(bool)*this`

Definition at line 1796 of file functional.

5.455.2.3 `template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (const function<
_Res(_ArgTypes...)> & __x)`

Function copy constructor.

Parameters

`x` A function object with identical call signature.

Postcondition

`(bool)*this == (bool)x`

The newly-created function contains a copy of the target of `x` (if it has one).

Definition at line 2033 of file functional.

5.455.2.4 `template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::function (function< _Res(_ArgTypes...)> &&
__x) [inline]`

Function move constructor.

Parameters

x A function object rvalue with identical call signature.

The newly-created function contains the target of *x* (if it has one).

Definition at line 1815 of file functional.

```
5.455.2.5 template<typename _Res , typename... _ArgTypes>
            template<typename _Function > std::function<
              _Res(_ArgTypes...)>::function ( _Function __f, typename enable_if<
                !is_integral< _Function >::value, _Useless >::type = _Useless() )
```

Builds a function that targets a copy of the incoming function object.

Parameters

f A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to *Res*.

The newly-created function object will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2047 of file functional.

5.455.3 Member Function Documentation

```
5.455.3.1 template<typename _Res , typename... _ArgTypes> std::function<
              _Res(_ArgTypes...)>::operator bool ( ) const [inline,
              explicit]
```

Determine if the function wrapper has a target.

Returns

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 1976 of file functional.

5.455.3.2 `template<typename _Res , typename... _ArgTypes> _Res
std::function< _Res(_ArgTypes...)>::operator() (_ArgTypes...
__args) const`

Invokes the function targeted by `*this`.

Returns

the result of the target.

Exceptions

bad_function_call when `!(bool)*this`

The function call operator invokes the target function object stored by `this`.

Definition at line 2065 of file functional.

5.455.3.3 `template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= (function<
_Res(_ArgTypes...)> && __x) [inline]`

Function move-assignment operator.

Parameters

x A function rvalue with identical call signature.

Returns

`*this`

The target of `x` is moved to `*this`. If `x` has no target, then `*this` will be empty.

If `x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1875 of file functional.

5.455.3.4 `template<typename _Res , typename... _ArgTypes>
template<typename _Function > enable_if<!is_integral<_
Function>::value, function&>::type std::function<
_Res(_ArgTypes...)>::operator= (_Function && __f) [inline]`

Function assignment to a new target.

Parameters

f A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res.

Returns

*this

This function object wrapper will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 1918 of file functional.

```
5.455.3.5 template<typename _Res , typename... _ArgTypes>
template<typename _Functor > enable_if<!is_integral<_Functor>::value, function<>::type std::function<
_Res(_ArgTypes...)>::operator= ( reference_wrapper< _Functor >
__f ) [inline]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 1927 of file functional.

```
5.455.3.6 template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= ( const function<
_Res(_ArgTypes...)> & __x ) [inline]
```

Function assignment operator.

Parameters

x A function with identical call signature.

Postcondition

`(bool)*this == (bool)x`

Returns

*this

The target of *x* is copied to **this*. If *x* has no target, then **this* will be empty.

If *x* targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1857 of file functional.

```
5.455.3.7  template<typename _Res , typename... _ArgTypes> function&
            std::function< _Res(_ArgTypes...)>::operator= ( nullptr_t )
            [inline]
```

Function assignment to zero.

Postcondition

!(bool)*this

Returns

*this

The target of **this* is deallocated, leaving it empty.

Definition at line 1889 of file functional.

```
5.455.3.8  template<typename _Res , typename... _ArgTypes> void
            std::function< _Res(_ArgTypes...)>::swap ( function<
            _Res(_ArgTypes...)> & __x ) [inline]
```

Swap the targets of two function objects.

Parameters

f A function with identical call signature.

Swap the targets of *this* function object and *f*. This function will not throw an exception.

Definition at line 1942 of file functional.

```
5.455.3.9  template<typename _Res , typename... _ArgTypes>
            template<typename _Functor > const _Functor * std::function<
            _Res(_ArgTypes...)>::target ( ) const
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2111 of file functional.

5.455.3.10 `template<typename _Res , typename... _ArgTypes>
 template<typename _Functor > _Functor * std::function<
 _Res(_ArgTypes...)>::target ()`

Access the stored target function object.

Returns

Returns a pointer to the stored target function object, if `typeid(Functor).equals(target_type())`; otherwise, a NULL pointer.

This function will not throw an exception.

Definition at line 2092 of file functional.

5.455.3.11 `template<typename _Res , typename... _ArgTypes> const type_info
 & std::function< _Res(_ArgTypes...)>::target_type () const`

Determine the type of the target of this function object wrapper.

Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

Definition at line 2076 of file functional.

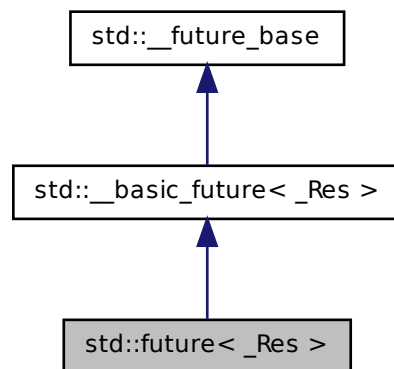
The documentation for this class was generated from the following file:

- [functional](#)

5.456 `std::future< _Res >` Class Template Reference

Primary template for future.

Inheritance diagram for std::future< _Res >:



Public Member Functions

- [future](#) ([future](#) &&__uf)
- **future** (const [future](#) &)
- _Res [get](#) ()
- [future](#) & **operator=** (const [future](#) &)
- [future](#) & **operator=** ([future](#) &&__fut)
- bool **valid** () const
- void **wait** () const
- template<typename _Rep , typename _Period >
bool **wait_for** (const [chrono::duration](#)< _Rep, _Period > &__rel) const
- template<typename _Clock , typename _Duration >
bool **wait_until** (const [chrono::time_point](#)< _Clock, _Duration > &__abs)
const

Protected Types

- typedef [__future_base::Result](#)< _Res > & __result_type

Protected Member Functions

- `__result_type _M_get_result()`
- `void _M_swap (__basic_future &__that)`

Friends

- `template<typename _Fn, typename... _Args>
future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&
_Args &&...)`
- `class packaged_task`
- `class promise< _Res >`

5.456.1 Detailed Description

`template<typename _Res> class std::future< _Res >`

Primary template for future.

Definition at line 559 of file future.

5.456.2 Constructor & Destructor Documentation

5.456.2.1 `template<typename _Res> std::future< _Res >::future (future<
_Res > && __uf) [inline]`

Move constructor.

Definition at line 577 of file future.

5.456.3 Member Function Documentation

5.456.3.1 `template<typename _Res> __result_type std::__basic_future< _Res
>::__M_get_result () [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

5.456.3.2 template<typename _Res > _Res std::future< _Res >::get () [inline]

Retrieving the value.

Definition at line 591 of file future.

References std::__basic_future< _Res >::M_get_result().

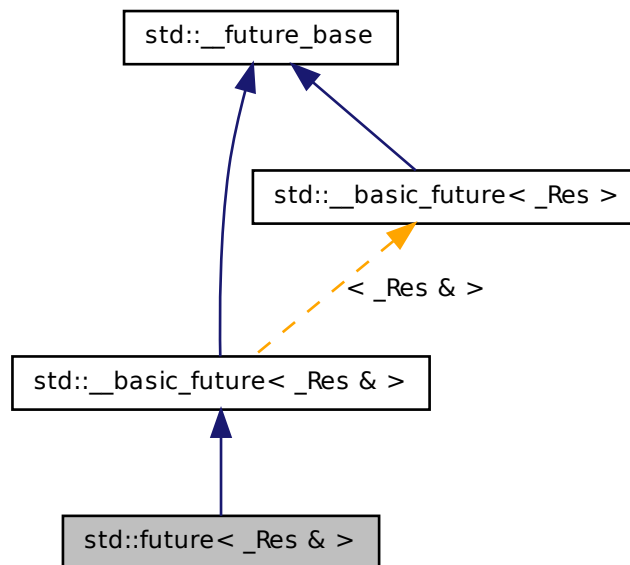
The documentation for this class was generated from the following file:

- [future](#)

5.457 std::future< _Res & > Class Template Reference

Partial specialization for future<R&>

Inheritance diagram for std::future< _Res & >:



Public Member Functions

- `future` (`future` &&__uf)
- `future` (const `future` &)
- `_Res` & `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&__fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- bool `wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs) const

Protected Types

- typedef `__future_base::_Result`< _Res & > & `__result_type`

Protected Member Functions

- `__result_type` `_M_get_result` ()
- void `_M_swap` (`__basic_future` &__that)

Friends

- template<typename _Fn, typename... _Args>
`future`< typename result_of< _Fn(_Args...)>::type > `async` (launch, _Fn &&, _Args &&...)
- class `packaged_task`
- class `promise`< _Res & >

5.457.1 Detailed Description

template<typename _Res> class std::future< _Res & >

Partial specialization for future<R&>

Definition at line 600 of file future.

5.457.2 Constructor & Destructor Documentation

5.457.2.1 `template<typename _Res > std::future< _Res & >::future (future< _Res & > && __uf) [inline]`

Move constructor.

Definition at line 618 of file `future`.

5.457.3 Member Function Documentation

5.457.3.1 `__result_type std::__basic_future< _Res & >::M_get_result () [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file `future`.

References `std::rethrow_exception()`.

5.457.3.2 `template<typename _Res > _Res& std::future< _Res & >::get () [inline]`

Retrieving the value.

Definition at line 632 of file `future`.

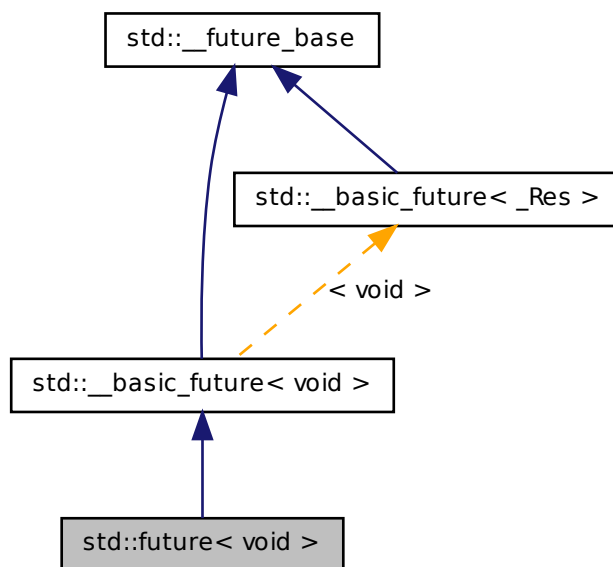
The documentation for this class was generated from the following file:

- [future](#)

5.458 `std::future< void >` Class Template Reference

Explicit specialization for [future<void>](#)

Inheritance diagram for `std::future< void >`:



Public Member Functions

- `future` (`future` &&__uf)
- `future` (const `future` &)
- void `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&__fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- bool `wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs) const

Protected Types

- typedef `__future_base::_Result`< void > & `__result_type`

Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap (__basic_future &__that)`

Friends

- `template<typename _Fn, typename... _Args>
future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&
_Args &&...)`
- `class packaged_task`
- `class promise< void >`

5.458.1 Detailed Description

`template<> class std::future< void >`

Explicit specialization for `future<void>`

Definition at line 641 of file future.

5.458.2 Constructor & Destructor Documentation

5.458.2.1 `std::future< void >::future (future< void > && __uf)
[inline]`

Move constructor.

Definition at line 659 of file future.

5.458.3 Member Function Documentation

5.458.3.1 `__result_type std::__basic_future< void >::_M_get_result ()
[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

5.458.3.2 `void std::future< void >::get () [inline]`

Retrieving the value.

Definition at line 673 of file future.

References `std::__basic_future< _Res >::_M_get_result()`.

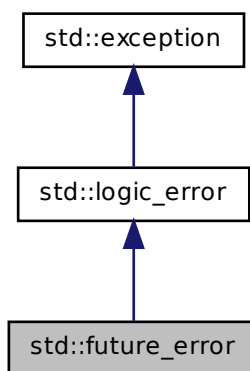
The documentation for this class was generated from the following file:

- [future](#)

5.459 std::future_error Class Reference

Exception type thrown by futures.

Inheritance diagram for `std::future_error`:



Public Member Functions

- **future_error** ([error_code](#) __ec)
- const [error_code](#) & **code** () const throw ()
- virtual const char * [what](#) () const throw ()

5.459.1 Detailed Description

Exception type thrown by futures.

Definition at line 85 of file future.

5.459.2 Member Function Documentation

5.459.2.1 `virtual const char* std::future_error::what () const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic_error](#).

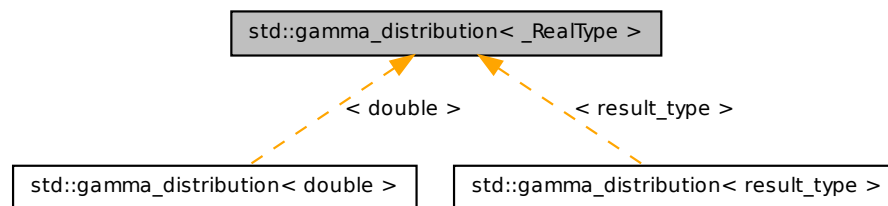
The documentation for this class was generated from the following file:

- [future](#)

5.460 `std::gamma_distribution<_RealType>` Class Template Reference

A gamma continuous distribution for random numbers.

Inheritance diagram for `std::gamma_distribution<_RealType>`:



Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- [gamma_distribution](#) ([_RealType](#) __alpha_val=[_RealType](#)(1), [_RealType](#) __beta_val=[_RealType](#)(1))
- [gamma_distribution](#) (const [param_type](#) &__p)
- [_RealType](#) [alpha](#) () const
- [_RealType](#) [beta](#) () const
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- template<typename [_UniformRandomNumberGenerator](#) >
[result_type](#) [operator\(\)](#) ([_UniformRandomNumberGenerator](#) &__urng, const [param_type](#) &__p)
- template<typename [_UniformRandomNumberGenerator](#) >
[result_type](#) [operator\(\)](#) ([_UniformRandomNumberGenerator](#) &__urng)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

Friends

- template<typename [_RealType1](#) , typename [_CharT](#) , typename [_Traits](#) >
[std::basic_ostream](#)< [_CharT](#), [_Traits](#) > & [operator<<](#) ([std::basic_ostream](#)< [_CharT](#), [_Traits](#) > &, const [std::gamma_distribution](#)< [_RealType1](#) > &)
- template<typename [_RealType1](#) >
bool [operator==](#) (const [std::gamma_distribution](#)< [_RealType1](#) > &__d1, const [std::gamma_distribution](#)< [_RealType1](#) > &__d2)
- template<typename [_RealType1](#) , typename [_CharT](#) , typename [_Traits](#) >
[std::basic_istream](#)< [_CharT](#), [_Traits](#) > & [operator>>](#) ([std::basic_istream](#)< [_CharT](#), [_Traits](#) > &, [std::gamma_distribution](#)< [_RealType1](#) > &)

5.460.1 Detailed Description

template<typename [_RealType](#) = double> class [std::gamma_distribution](#)< [_RealType](#) >

A gamma continuous distribution for random numbers. The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2350 of file random.h.

5.460.2 Member Typedef Documentation

5.460.2.1 `template<typename _RealType = double> typedef _RealType
std::gamma_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2357 of file `random.h`.

5.460.3 Constructor & Destructor Documentation

5.460.3.1 `template<typename _RealType = double> std::gamma_distribution<
_RealType>::gamma_distribution (_RealType __alpha_val
= _RealType(1), _RealType __beta_val = _RealType(1))
[inline, explicit]`

Constructs a gamma distribution with parameters α and β .

Definition at line 2402 of file `random.h`.

5.460.4 Member Function Documentation

5.460.4.1 `template<typename _RealType = double> _RealType
std::gamma_distribution<_RealType>::alpha () const
[inline]`

Returns the α of the distribution.

Definition at line 2423 of file `random.h`.

5.460.4.2 `template<typename _RealType = double> _RealType
std::gamma_distribution<_RealType>::beta () const [inline]`

Returns the β of the distribution.

Definition at line 2430 of file `random.h`.

5.460.4.3 `template<typename _RealType = double> result_type
std::gamma_distribution<_RealType>::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2459 of file `random.h`.

Referenced by `std::gamma_distribution<result_type>::max()`.

5.460.4.4 `template<typename _RealType = double> result_type
std::gamma_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2452 of file random.h.

5.460.4.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::gamma_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2467 of file random.h.

Referenced by `std::gamma_distribution< result_type >::operator()()`.

5.460.4.6 `template<typename _RealType > template<typename
_UniformRandomNumberGenerator > gamma_distribution<
_RealType >::result_type std::gamma_distribution< _RealType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables"
ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2009 of file random.tcc.

References `std::log()`, and `std::pow()`.

5.460.4.7 `template<typename _RealType = double> void
std::gamma_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 2445 of file random.h.

5.460.4.8 `template<typename _RealType = double> param_type
std::gamma_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2437 of file random.h.

Referenced by std::gamma_distribution< result_type >::operator()().

5.460.4.9 `template<typename _RealType = double> void
std::gamma_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2416 of file random.h.

Referenced by std::negative_binomial_distribution< _IntType >::reset(), std::student_t_distribution< _RealType >::reset(), std::fisher_f_distribution< _RealType >::reset(), and std::chi_squared_distribution< _RealType >::reset().

5.460.5 Friends And Related Function Documentation

5.460.5.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::gamma_distribution< _RealType1 > &) [friend]`

Inserts a gamma_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A gamma_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.460.5.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::gamma_distribution<
_RealType1 > & __d1, const std::gamma_distribution< _RealType1
> & __d2) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2482 of file random.h.

5.460.5.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::gamma_distribution< _RealType1 > &) [friend]`

Extracts a gamma_distribution random number distribution __x from the input stream __is.

Parameters

`__is` An input stream.

`__x` A gamma_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.461 `std::gamma_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef `gamma_distribution< _RealType >` `distribution_type`

Public Member Functions

- `param_type` (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)

- `_RealType alpha () const`
- `_RealType beta () const`

Friends

- class `gamma_distribution< _RealType >`
- bool `operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.461.1 Detailed Description

`template<typename _RealType = double> struct std::gamma_distribution< _RealType >::param_type`

Parameter type.

Definition at line 2359 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.462 `std::geometric_distribution< _IntType >` Class Template Reference

A discrete geometric random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- `geometric_distribution` (double __p=0.5)
- `geometric_distribution` (const [param_type](#) &__p)
- [result_type](#) `max () const`

- [result_type min](#) () const
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng, const
param_type &__p)`
- `double p () const`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

5.462.1 Detailed Description

`template<typename _IntType = int> class std::geometric_distribution< _IntType >`

A discrete geometric random number distribution. The formula for the geometric probability density function is $p(i|p) = (1 - p)p^{i-1}$ where p is the parameter of the distribution.

Definition at line 3619 of file random.h.

5.462.2 Member Typedef Documentation

5.462.2.1 `template<typename _IntType = int> typedef _IntType
std::geometric_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3626 of file random.h.

5.462.3 Member Function Documentation

5.462.3.1 `template<typename _IntType = int> result_type
std::geometric_distribution< _IntType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3712 of file random.h.

5.462.3.2 `template<typename _IntType = int> result_type
std::geometric_distribution< _IntType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3705 of file random.h.

5.462.3.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::geometric_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3720 of file random.h.

References `std::geometric_distribution< _IntType >::operator()()`, and
`std::geometric_distribution< _IntType >::param()`.

Referenced by `std::geometric_distribution< _IntType >::operator()()`.

5.462.3.4 `template<typename _IntType = int> double
std::geometric_distribution< _IntType >::p () const [inline]`

Returns the distribution parameter `p`.

Definition at line 3683 of file random.h.

5.462.3.5 `template<typename _IntType = int> void std::geometric_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3698 of file random.h.

5.462.3.6 `template<typename _IntType = int> param_type
std::geometric_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 3690 of file random.h.

Referenced by `std::geometric_distribution< _IntType >::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.462.3.7 `template<typename _IntType = int> void
std::geometric_distribution< _IntType >::reset () [inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 3677 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.463 `std::geometric_distribution< _IntType >::param_type` Struct Reference

Public Types

- typedef `geometric_distribution< _IntType >` `distribution_type`

Public Member Functions

- `param_type` (double __p=0.5)
- double `p` () const

Friends

- class `geometric_distribution< _IntType >`
- bool `operator==(const param_type &__p1, const param_type &__p2)`

5.463.1 Detailed Description

`template<typename _IntType = int> struct std::geometric_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3628 of file random.h.

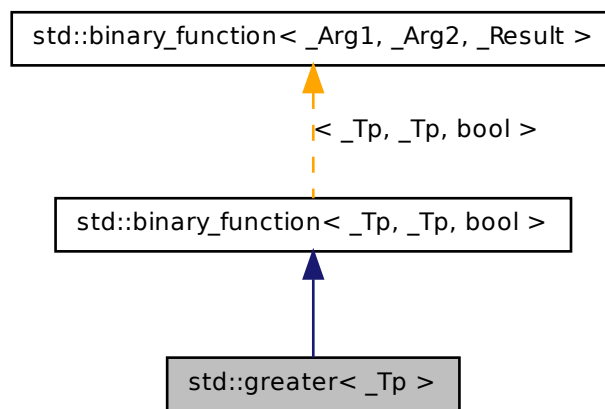
The documentation for this struct was generated from the following file:

- [random.h](#)

5.464 std::greater< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::greater< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.464.1 Detailed Description

`template<typename _Tp> struct std::greater< _Tp >`

One of the [comparison functors](#).

Definition at line 217 of file `stl_function.h`.

5.464.2 Member Typedef Documentation

5.464.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.464.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.464.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

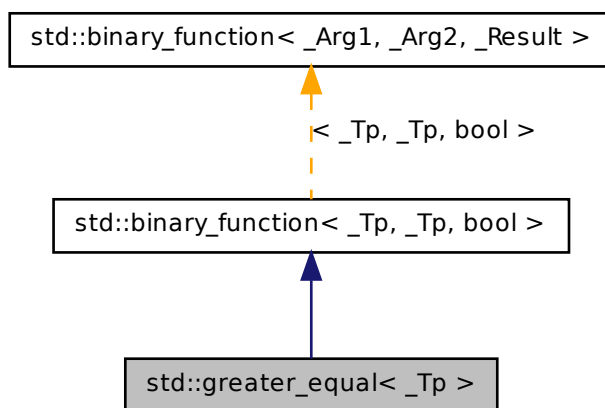
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.465 `std::greater_equal< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::greater_equal< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.465.1 Detailed Description

```
template<typename _Tp> struct std::greater_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 235 of file `stl_function.h`.

5.465.2 Member Typedef Documentation

5.465.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.465.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.465.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.466 std::gslice Class Reference

Class defining multi-dimensional subset of an array.

Public Member Functions

- [gslice](#) ()
- [gslice](#) (size_t, const [valarray](#)< size_t > &, const [valarray](#)< size_t > &)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size_t > [size](#) () const
- size_t [start](#) () const
- [valarray](#)< size_t > [stride](#) () const

Friends

- class `valarray`

5.466.1 Detailed Description

Class defining multi-dimensional subset of an array. The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 63 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

5.467 `std::gslice_array< _Tp >` Class Template Reference

Reference to multi-dimensional subset of an array.

Public Types

- typedef `_Tp value_type`

Public Member Functions

- [gslice_array](#) (const [gslice_array](#) &)
- template<class `_Dom` >
void **operator%=** (const `_Expr< _Dom, _Tp >` &) const
- void **operator%=** (const [valarray](#)< `_Tp` > &) const
- void **operator&=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >
void **operator&=** (const `_Expr< _Dom, _Tp >` &) const

- `template<class _Dom >`
`void operator*= (const _Expr< _Dom, _Tp > &) const`
- `void operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator+= (const _Expr< _Dom, _Tp > &) const`
- `void operator+= (const valarray< _Tp > &) const`
- `void operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator-= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void operator/= (const _Expr< _Dom, _Tp > &) const`
- `void operator/= (const valarray< _Tp > &) const`
- `void operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const _Tp &) const`
- `gslice_array & operator= (const gslice_array &)`
- `template<class _Dom >`
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator|= (const _Expr< _Dom, _Tp > &) const`

Friends

- `class valarray< _Tp >`

5.467.1 Detailed Description

`template<typename _Tp> class std::gslice_array< _Tp >`

Reference to multi-dimensional subset of an array. A [gslice_array](#) is a reference to the actual elements of an array specified by a [gslice](#). The way to get a [gslice_array](#) is to call `operator[]`([gslice](#)) on a [valarray](#). The returned [gslice_array](#) then permits carrying operations out on the referenced subset of elements in the original [valarray](#). For example, `operator+=(valarray)` will add values to the subset of elements in the underlying [valarray](#) this [gslice_array](#) refers to.

5.468 `std::has_nothrow_copy_assign< _Tp >` Struct Template Reference 2495

Parameters

Tp Element type.

Definition at line 59 of file `gslice_array.h`.

The documentation for this class was generated from the following file:

- [gslice_array.h](#)

5.468 `std::has_nothrow_copy_assign< _Tp >` Struct Template Reference

[has_nothrow_copy_assign](#)

Inherits `integral_constant< bool, __has_nothrow_assign(_Tp)>`.

5.468.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_copy_assign< _Tp >
```

[has_nothrow_copy_assign](#)

Definition at line 297 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.469 `std::has_nothrow_copy_constructor< _Tp >` Struct Template Reference

[has_nothrow_copy_constructor](#)

Inherits `integral_constant< bool, __has_nothrow_copy(_Tp)>`.

5.469.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_copy_constructor< _Tp >
```

[has_nothrow_copy_constructor](#)

Definition at line 291 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.470 `std::has_nothrow_default_constructor< _Tp >` Struct Template Reference

[has_nothrow_default_constructor](#)

Inherits `integral_constant< bool, __has_nothrow_constructor(_Tp)>`.

5.470.1 Detailed Description

`template<typename _Tp> struct std::has_nothrow_default_constructor< _Tp >`

[has_nothrow_default_constructor](#)

Definition at line 285 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.471 `std::has_trivial_copy_assign< _Tp >` Struct Template Reference

[has_trivial_copy_assign](#)

Inherits `integral_constant< bool, __has_trivial_assign(_Tp)>`.

5.471.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_copy_assign< _Tp >`

[has_trivial_copy_assign](#)

Definition at line 273 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.472 std::has_trivial_copy_constructor< _Tp > Struct Template Reference 2497

5.472 std::has_trivial_copy_constructor< _Tp > Struct Template Reference

[has_trivial_copy_constructor](#)

Inherits `integral_constant< bool, __has_trivial_copy(_Tp)>`.

5.472.1 Detailed Description

```
template<typename _Tp> struct std::has_trivial_copy_constructor< _Tp >
```

[has_trivial_copy_constructor](#)

Definition at line 267 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.473 std::has_trivial_default_constructor< _Tp > Struct Template Reference

[has_trivial_default_constructor](#)

Inherits `integral_constant< bool, __has_trivial_constructor(_Tp)>`.

5.473.1 Detailed Description

```
template<typename _Tp> struct std::has_trivial_default_constructor< _Tp >
```

[has_trivial_default_constructor](#)

Definition at line 261 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.474 std::has_trivial_destructor< _Tp > Struct Tem- plate Reference

[has_trivial_destructor](#)

Inherits `integral_constant< bool, __has_trivial_destructor(_Tp)>`.

5.474.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_destructor< _Tp >`

[has_trivial_destructor](#)

Definition at line 279 of file `type_traits`.

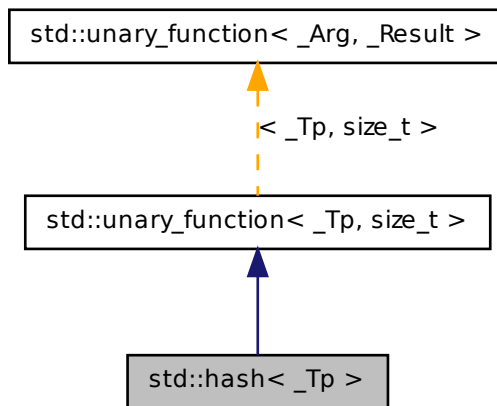
The documentation for this struct was generated from the following file:

- [type_traits](#)

5.475 `std::hash< _Tp >` Struct Template Reference

Primary class template hash.

Inheritance diagram for `std::hash< _Tp >`:



Public Types

- `typedef _Tp` [argument_type](#)

- typedef size_t [result_type](#)

Public Member Functions

- size_t **operator()** (_Tp __val) const
- template<>
size_t **operator()** (double __val) const
- template<>
size_t **operator()** (float __val) const
- template<>
size_t **operator()** (unsigned long long __val) const
- template<>
size_t **operator()** (unsigned long __val) const
- template<>
size_t **operator()** (unsigned int __val) const
- template<>
size_t **operator()** (unsigned short __val) const
- template<>
size_t **operator()** (long long __val) const
- template<>
size_t **operator()** (long __val) const
- template<>
size_t **operator()** (int __val) const
- template<>
size_t **operator()** (short __val) const
- template<>
size_t **operator()** (char32_t __val) const
- template<>
size_t **operator()** (char16_t __val) const
- template<>
size_t **operator()** (wchar_t __val) const
- template<>
size_t **operator()** (unsigned char __val) const
- template<>
size_t **operator()** (signed char __val) const
- template<>
size_t **operator()** (char __val) const
- template<>
size_t **operator()** (bool __val) const

5.475.1 Detailed Description

template<typename _Tp> struct std::hash< _Tp >

Primary class template hash.

Definition at line 50 of file functional_hash.h.

5.475.2 Member Typedef Documentation

**5.475.2.1 typedef _Tp std::unary_function< _Tp , size_t >::argument_type
[inherited]**

argument_type is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

**5.475.2.2 typedef size_t std::unary_function< _Tp , size_t >::result_type
[inherited]**

result_type is the return type

Definition at line 105 of file stl_function.h.

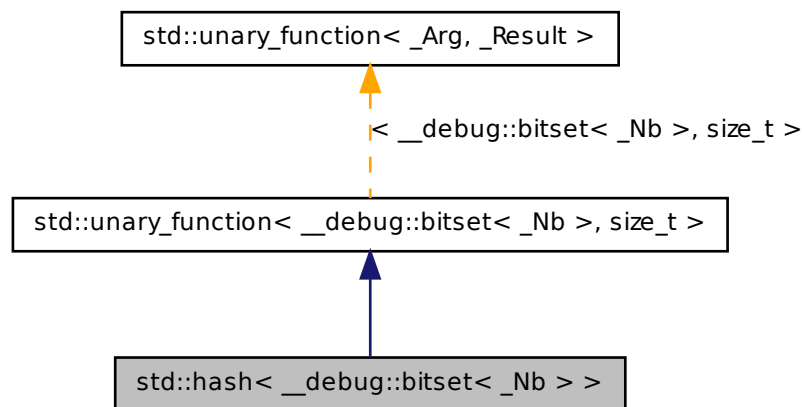
The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

5.476 std::hash< __debug::bitset< _Nb > > Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inheritance diagram for std::hash< __debug::bitset< _Nb > >:



Public Types

- typedef `__debug::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __debug::bitset< _Nb > &__b) const`

5.476.1 Detailed Description

`template<size_t _Nb> struct std::hash< __debug::bitset< _Nb > >`

`std::hash` specialization for `bitset`.

Definition at line 387 of file `debug/bitset`.

5.476.2 Member Typedef Documentation

5.476.2.1 `typedef __debug::bitset< _Nb > std::unary_function<
__debug::bitset< _Nb > , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.476.2.2 `typedef size_t std::unary_function< __debug::bitset< _Nb > , size_t
>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

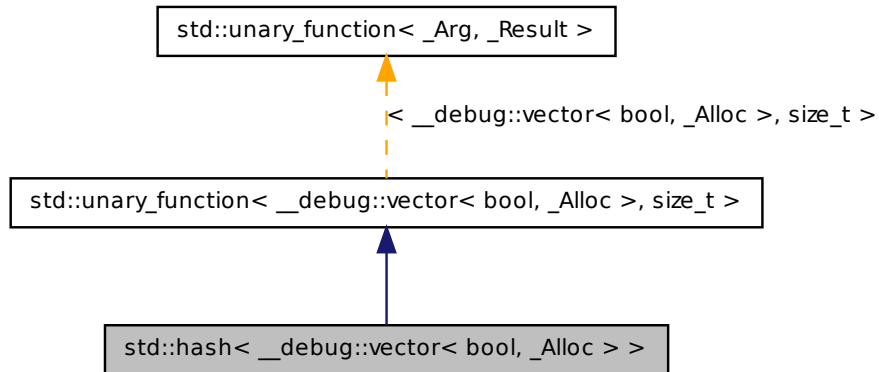
- [debug/bitset](#)

5.477 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

5.477 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

Inheritance diagram for `std::hash< __debug::vector< bool, _Alloc > >`:



Public Types

- typedef `__debug::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __debug::vector< bool, _Alloc > &__b) const`

5.477.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __debug::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 583 of file `debug/vector`.

5.477.2 Member Typedef Documentation

5.477.2.1 `typedef __debug::vector< bool, _Alloc > std::unary_function<
__debug::vector< bool, _Alloc > , size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.477.2.2 `typedef size_t std::unary_function< __debug::vector< bool, _Alloc >
, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

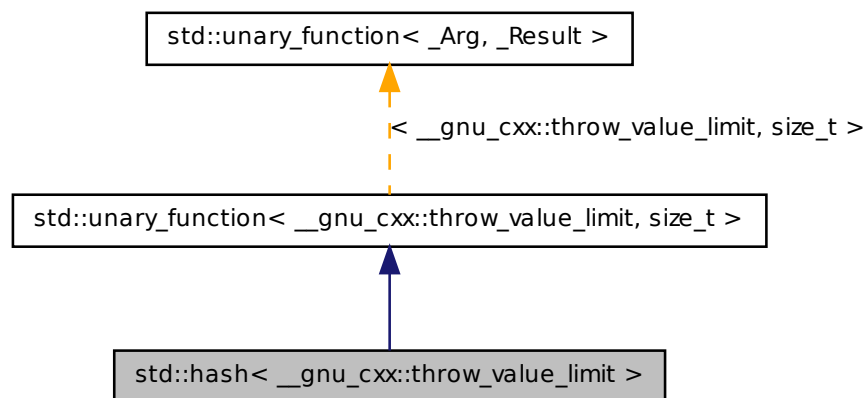
- [debug/vector](#)

5.478 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Explicit specialization of [std::hash](#) for [__gnu_cxx::throw_value_limit](#).

5.478 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference 2505

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_limit &__val) const`

5.478.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::throw_value_limit >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 734 of file `throw_allocator.h`.

5.478.2 Member Typedef Documentation

5.478.2.1 `typedef __gnu_cxx::throw_value_limit std::unary_function<
__gnu_cxx::throw_value_limit, size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.478.2.2 `typedef size_t std::unary_function< __gnu_cxx::throw_value_limit,
size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

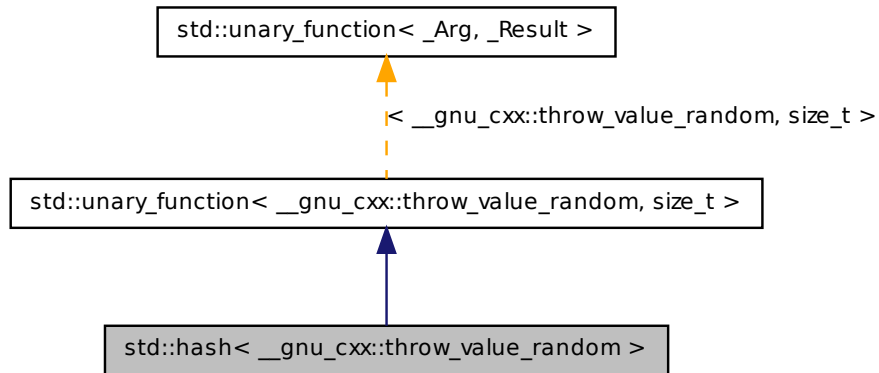
- [throw_allocator.h](#)

5.479 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Explicit specialization of [std::hash](#) for [__gnu_cxx::throw_value_limit](#).

5.479 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_random &__val) const`

5.479.1 Detailed Description

template<> struct `std::hash< __gnu_cxx::throw_value_random >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 748 of file `throw_allocator.h`.

5.479.2 Member Typedef Documentation

5.479.2.1 `typedef __gnu_cxx::throw_value_random std::unary_function<
__gnu_cxx::throw_value_random , size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.479.2.2 `typedef size_t std::unary_function< __gnu_-
cxx::throw_value_random , size_t >::result_type
[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

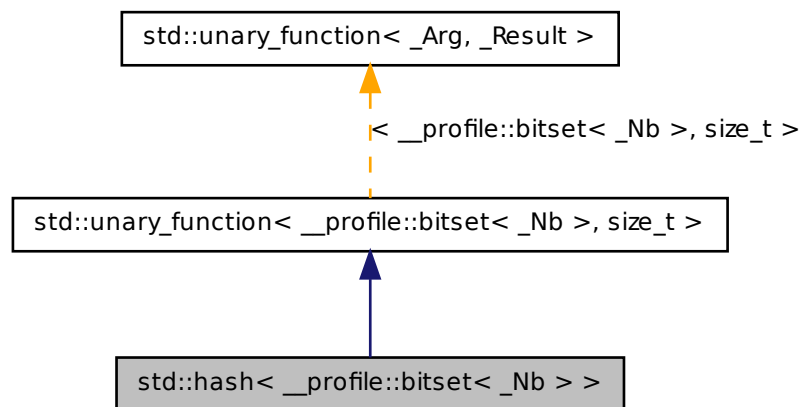
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.480 `std::hash< __profile::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inheritance diagram for `std::hash< __profile::bitset< _Nb > >`:



Public Types

- typedef `__profile::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __profile::bitset< _Nb > &__b) const`

5.480.1 Detailed Description

`template<size_t _Nb> struct std::hash< __profile::bitset< _Nb > >`

`std::hash` specialization for `bitset`.

Definition at line 361 of file `profile/bitset`.

5.480.2 Member Typedef Documentation

5.480.2.1 `typedef __profile::bitset< _Nb > std::unary_function<
__profile::bitset< _Nb >, size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.480.2.2 `typedef size_t std::unary_function< __profile::bitset< _Nb >, size_t
>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

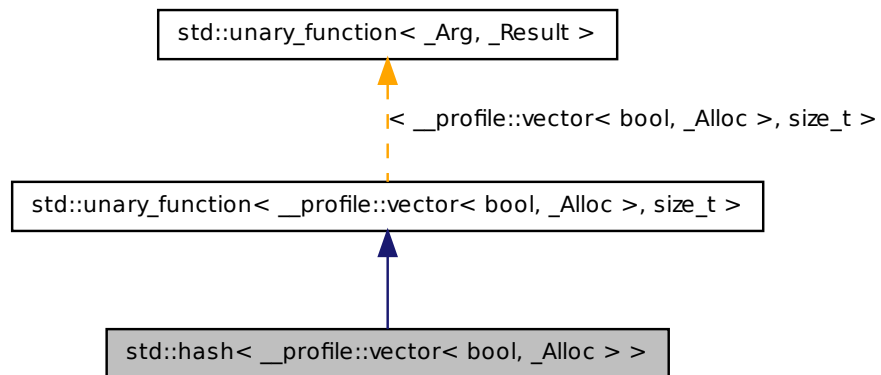
- [profile/bitset](#)

5.481 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

5.481 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

Inheritance diagram for `std::hash< __profile::vector< bool, _Alloc > >`:



Public Types

- typedef `__profile::vector< bool, _Alloc >` [argument_type](#)
- typedef `size_t` [result_type](#)

Public Member Functions

- `size_t operator() (const __profile::vector< bool, _Alloc > &__b) const`

5.481.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __profile::vector< bool, _Alloc >  
>
```

[std::hash](#) specialization for `vector<bool>`.

Definition at line 507 of file `profile/vector`.

5.481.2 Member Typedef Documentation

5.481.2.1 `typedef __profile::vector< bool, _Alloc > std::unary_function<
__profile::vector< bool, _Alloc > , size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.481.2.2 `typedef size_t std::unary_function< __profile::vector< bool, _Alloc
> , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

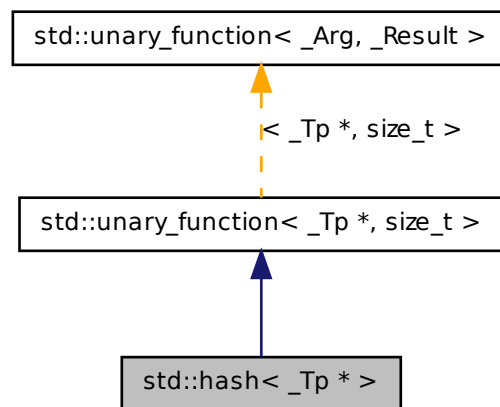
The documentation for this struct was generated from the following file:

- [profile/vector](#)

5.482 `std::hash< _Tp * >` Struct Template Reference

Partial specializations for pointer types.

Inheritance diagram for std::hash< _Tp * >:



Public Types

- typedef `_Tp *` [argument_type](#)
- typedef `size_t` [result_type](#)

Public Member Functions

- `size_t operator() (_Tp *__p) const`

5.482.1 Detailed Description

template<typename _Tp> struct std::hash< _Tp * >

Partial specializations for pointer types.

Definition at line 58 of file `functional_hash.h`.

5.482.2 Member Typedef Documentation

5.482.2.1 `typedef _Tp * std::unary_function< _Tp *, size_t >::argument_type` [*inherited*]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.482.2.2 `typedef size_t std::unary_function< _Tp *, size_t >::result_type` [*inherited*]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

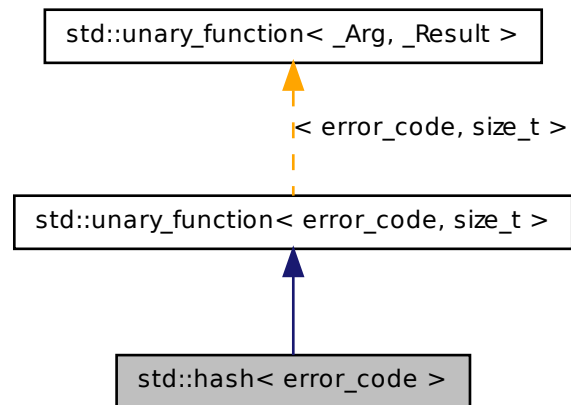
The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

5.483 `std::hash< error_code >` Struct Template Reference

[std::hash](#) specialization for [error_code](#).

Inheritance diagram for std::hash< error_code >:



Public Types

- typedef `error_code` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const error_code &__e) const`

5.483.1 Detailed Description

`template<> struct std::hash< error_code >`

`std::hash` specialization for `error_code`.

Definition at line 352 of file `system_error`.

5.483.2 Member Typedef Documentation

5.483.2.1 `typedef error_code std::unary_function< error_code , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.483.2.2 `typedef size_t std::unary_function< error_code , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

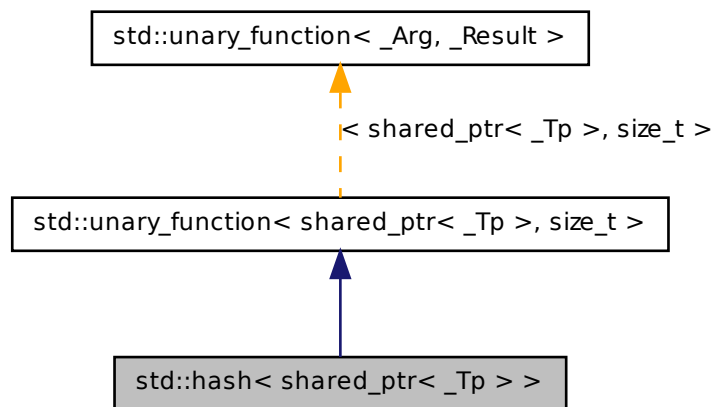
The documentation for this struct was generated from the following file:

- [system_error](#)

5.484 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

`std::hash` specialization for [shared_ptr](#).

Inheritance diagram for std::hash< shared_ptr< _Tp > >:



Public Types

- typedef `shared_ptr< _Tp >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const shared_ptr< _Tp > &__s) const`

5.484.1 Detailed Description

`template<typename _Tp> struct std::hash< shared_ptr< _Tp > >`

`std::hash` specialization for `shared_ptr`.

Definition at line 519 of file `shared_ptr.h`.

5.484.2 Member Typedef Documentation

5.484.2.1 `typedef shared_ptr< _Tp > std::unary_function< shared_ptr< _Tp >, size_t>::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.484.2.2 `typedef size_t std::unary_function< shared_ptr< _Tp >, size_t>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

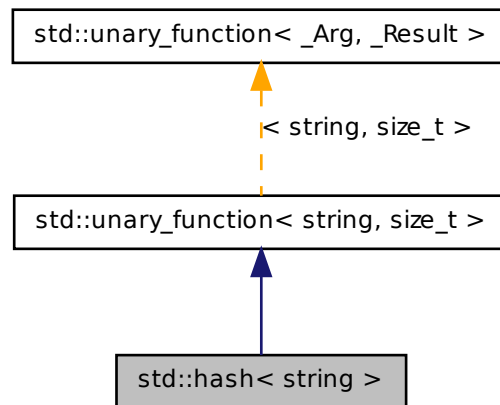
The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

5.485 `std::hash< string >` Struct Template Reference

[std::hash](#) specialization for string.

Inheritance diagram for std::hash< string >:



Public Types

- typedef [string](#) `argument_type`
- typedef `size_t` [result_type](#)

Public Member Functions

- `size_t operator() (const string &__s) const`

5.485.1 Detailed Description

template<> struct std::hash< string >

[std::hash](#) specialization for string.

Definition at line 2920 of file `basic_string.h`.

5.485.2 Member Typedef Documentation

5.485.2.1 `typedef string std::unary_function< string , size_t >::argument_type` `[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.485.2.2 `typedef size_t std::unary_function< string , size_t >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

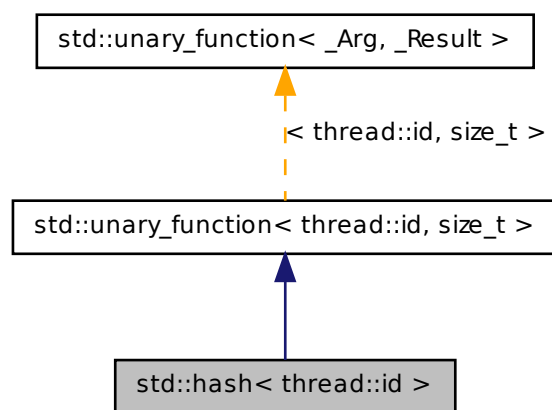
The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.486 `std::hash< thread::id >` Struct Template Reference

[std::hash](#) specialization for [thread::id](#).

Inheritance diagram for std::hash< thread::id >:



Public Types

- typedef `thread::id` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const thread::id &__id) const`

5.486.1 Detailed Description

`template<> struct std::hash< thread::id >`

`std::hash` specialization for `thread::id`.

Definition at line 225 of file `thread`.

5.486.2 Member Typedef Documentation

5.486.2.1 `typedef thread::id std::unary_function< thread::id , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.486.2.2 `typedef size_t std::unary_function< thread::id , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

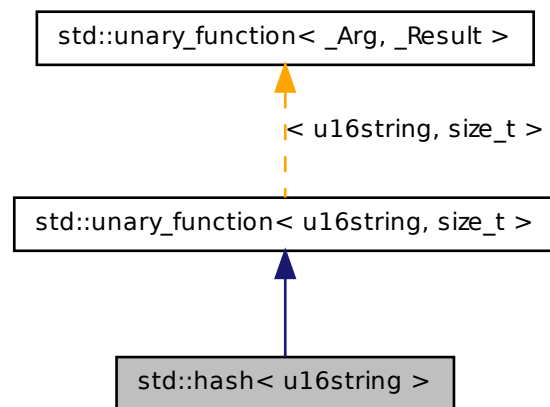
The documentation for this struct was generated from the following file:

- [thread](#)

5.487 `std::hash< u16string >` Struct Template Reference

[std::hash](#) specialization for `u16string`.

Inheritance diagram for std::hash< u16string >:



Public Types

- typedef `u16string` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const u16string &__s) const`

5.487.1 Detailed Description

template<> struct std::hash< u16string >

`std::hash` specialization for `u16string`.

Definition at line 2945 of file `basic_string.h`.

5.487.2 Member Typedef Documentation

5.487.2.1 `typedef u16string std::unary_function< u16string , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.487.2.2 `typedef size_t std::unary_function< u16string , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

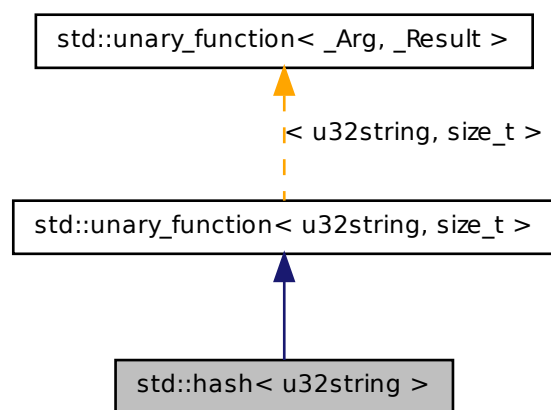
The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.488 `std::hash< u32string >` Struct Template Reference

[std::hash](#) specialization for `u32string`.

Inheritance diagram for std::hash< u32string >:



Public Types

- typedef `u32string` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const u32string &__s) const`

5.488.1 Detailed Description

`template<> struct std::hash< u32string >`

`std::hash` specialization for `u32string`.

Definition at line 2956 of file `basic_string.h`.

5.488.2 Member Typedef Documentation

5.488.2.1 `typedef u32string std::unary_function< u32string , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.488.2.2 `typedef size_t std::unary_function< u32string , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

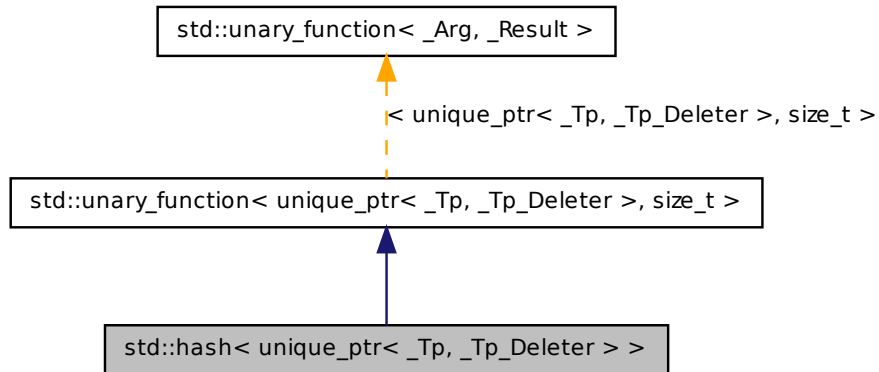
- [basic_string.h](#)

5.489 `std::hash< unique_ptr< _Tp, _Tp_Deleter > >` Struct Template Reference

[std::hash](#) specialization for [unique_ptr](#).

5.489 `std::hash< unique_ptr< _Tp, _Tp_Deleter > >` Struct Template Reference

Inheritance diagram for `std::hash< unique_ptr< _Tp, _Tp_Deleter > >`:



Public Types

- typedef `unique_ptr< _Tp, _Tp_Deleter >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const unique_ptr< _Tp, _Tp_Deleter > &__u) const`

5.489.1 Detailed Description

`template<typename _Tp, typename _Tp_Deleter> struct std::hash< unique_ptr< _Tp, _Tp_Deleter > >`

`std::hash` specialization for `unique_ptr`.

Definition at line 449 of file `unique_ptr.h`.

5.489.2 Member Typedef Documentation

5.489.2.1 `typedef unique_ptr< _Tp, _Tp_Deleter > std::unary_function< unique_ptr< _Tp, _Tp_Deleter > , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.489.2.2 `typedef size_t std::unary_function< unique_ptr< _Tp, _Tp_Deleter > , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

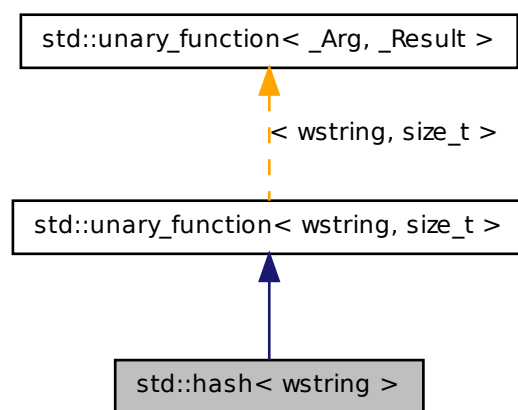
The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

5.490 `std::hash< wstring >` Struct Template Reference

[std::hash](#) specialization for `wstring`.

Inheritance diagram for std::hash< wstring >:



Public Types

- typedef `wstring` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const wstring &__s) const`

5.490.1 Detailed Description

`template<> struct std::hash< wstring >`

`std::hash` specialization for `wstring`.

Definition at line 2931 of file `basic_string.h`.

5.490.2 Member Typedef Documentation

5.490.2.1 `typedef wstring std::unary_function< wstring , size_t >::argument_type` `[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.490.2.2 `typedef size_t std::unary_function< wstring , size_t >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

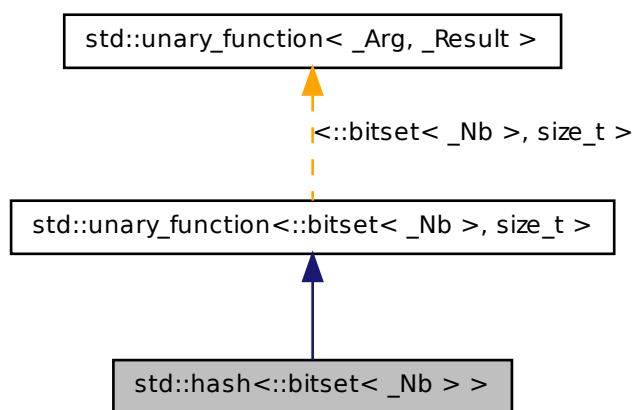
The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.491 `std::hash<::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inheritance diagram for std::hash<::bitset< _Nb > >:



Public Types

- typedef `::bitset< _Nb >` [argument_type](#)
- typedef `size_t` [result_type](#)

Public Member Functions

- `size_t operator() (const ::bitset< _Nb > &__b) const`

5.491.1 Detailed Description

`template<size_t _Nb> struct std::hash<::bitset< _Nb > >`

[std::hash](#) specialization for `bitset`.

Definition at line 1497 of file `bitset`.

5.491.2 Member Typedef Documentation

5.491.2.1 `typedef ::bitset< _Nb > std::unary_function< ::bitset< _Nb > ,
size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.491.2.2 `typedef size_t std::unary_function< ::bitset< _Nb > , size_t
>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

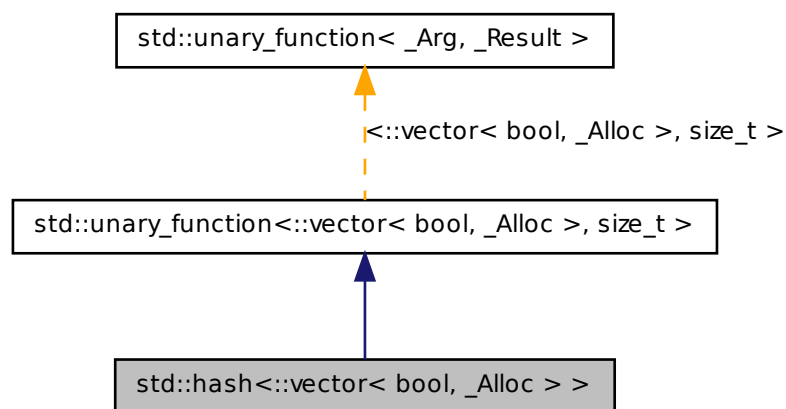
The documentation for this struct was generated from the following file:

- [bitset](#)

5.492 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inheritance diagram for `std::hash<::vector< bool, _Alloc > >`:



Public Types

- typedef `::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const ::vector< bool, _Alloc > &__b) const`

5.492.1 Detailed Description

template<typename _Alloc> struct `std::hash<::vector< bool, _Alloc > >`

`std::hash` specialization for `vector<bool>`.

Definition at line 1040 of file `stl_bvector.h`.

5.492.2 Member Typedef Documentation

5.492.2.1 `typedef ::vector< bool, _Alloc > std::unary_function< ::vector< bool, _Alloc >, size_t>::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.492.2.2 `typedef size_t std::unary_function< ::vector< bool, _Alloc >, size_t>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following files:

- [stl_bvector.h](#)
- [vector.tcc](#)

5.493 `std::identity< _Tp >` Struct Template Reference

`identity`

Public Types

- `typedef _Tp type`

5.493.1 Detailed Description

`template<typename _Tp> struct std::identity< _Tp >`

`identity`

Definition at line 56 of file `move.h`.

The documentation for this struct was generated from the following file:

- [move.h](#)

5.494 `std::independent_bits_engine<` `_-` `RandomNumberEngine, __w, _UIntType` `>` Class Template Reference

Public Types

- `typedef _UIntType` `result_type`

Public Member Functions

- `independent_bits_engine` ()
- `independent_bits_engine` (const `_RandomNumberEngine` &__rne)
- `independent_bits_engine` (`result_type` __s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`
`independent_bits_engine` (`_Sseq` &__q)
- `independent_bits_engine` (`_RandomNumberEngine` &&__rne)
- `const _RandomNumberEngine` & `base` () const
- `void discard` (unsigned long long __z)
- `result_type max` () const
- `result_type min` () const
- `result_type operator()` ()
- `void seed` (`result_type` __s)
- `template<typename _Sseq >`
`void seed` (`_Sseq` &__q)
- `void seed` ()

Friends

- `bool operator==` (const `independent_bits_engine` &__lhs, const `independent_bits_engine` &__rhs)
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream< _CharT, _Traits > &__is`, `std::independent_bits_engine< _-`
`RandomNumberEngine, __w, _UIntType > &__x`)

5.494.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1010 of file random.h.

5.494.2 Member Typedef Documentation

5.494.2.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type`

The type of the generated random value.

Definition at line 1019 of file random.h.

5.494.3 Constructor & Destructor Documentation

5.494.3.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine () [inline]`

Constructs a default `independent_bits_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1026 of file random.h.

5.494.3.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (const _RandomNumberEngine & __rne) [inline, explicit]`

Copy constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

Parameters

rng An existing (base class) engine object.

**5.494 std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType
> Class Template Reference** 2537

Definition at line 1036 of file random.h.

5.494.3.3 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_-
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
(_RandomNumberEngine && __rne) [inline, explicit]`

Move constructs a independent_bits_engine engine.

Copies an existing base class random number generator.

Parameters

rng An existing (base class) engine object.

Definition at line 1046 of file random.h.

5.494.3.4 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_-
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
(result_type __s) [inline, explicit]`

Seed constructs a independent_bits_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

__s A seed value for the base class engine.

Definition at line 1056 of file random.h.

5.494.3.5 `template<typename _RandomNumberEngine, size_t
__w, typename _UIntType> template<typename _Sseq ,
typename = typename std::enable_if<!std::is_same<_Sseq,
independent_bits_engine>::value && !std::is_same<_Sseq,
RandomNumberEngine>::value> ::type> std::independent
bits_engine< _RandomNumberEngine, __w, _UIntType
>::independent_bits_engine (_Sseq & __q) [inline,
explicit]`

Generator construct a independent_bits_engine engine.

Parameters

__q A seed sequence.

Definition at line 1069 of file random.h.

5.494.4 Member Function Documentation

5.494.4.1 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> const _RandomNumberEngine&
std::independent_bits_engine< _RandomNumberEngine, __w,
_UIntType >::base () const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 1104 of file random.h.

5.494.4.2 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::discard (unsigned
long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 1131 of file random.h.

5.494.4.3 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::max () const
[inline]`

Gets the maximum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1122 of file random.h.

5.494.4.4 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::min () const
[inline]`

Gets the minimum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1113 of file random.h.

5.494.4.5 `template<typename _RandomNumberEngine, size_t __w, typename
_UIntType > independent_bits_engine< _RandomNumberEngine,
__w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()`

Gets the next value in the generated random number sequence.

Definition at line 719 of file random.tcc.

References std::log().

5.494.4.6 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> template<typename _Sseq > void
std::independent_bits_engine< _RandomNumberEngine, __w,
_UIntType >::seed (_Sseq & __q) [inline]`

Reseeds the independent_bits_engine object with the given seed sequence.

Parameters

`__q` A seed generator function.

Definition at line 1096 of file random.h.

5.494.4.7 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::seed (result_type __s
) [inline]`

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

Definition at line 1086 of file random.h.

5.494.4.8 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::seed () [inline]`

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

Definition at line 1078 of file random.h.

5.494.5 Friends And Related Function Documentation

5.494.5.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator==(const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __lhs, const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __rhs) [friend]`

Compares two `independent_bits_engine` random number generator objects of the same type for equality.

Parameters

`__lhs` A `independent_bits_engine` random number generator object.

`__rhs` Another `independent_bits_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1156 of file random.h.

5.494.5.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>>(std::basic_istream<_CharT, _Traits> & __is, std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __x) [friend]`

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `independent_bits_engine` random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1174 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.495 `std::indirect_array<_Tp>` Class Template Reference

Reference to arbitrary subset of an array.

Public Types

- `typedef _Tp value_type`

Public Member Functions

- `indirect_array` (const `indirect_array` &)
- `template<class _Dom >`
void **operator**%= (const `_Expr<_Dom, _Tp>` &) const
- void `operator`%= (const `valarray<_Tp>` &) const
- void `operator`&= (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **operator**&= (const `_Expr<_Dom, _Tp>` &) const
- `template<class _Dom >`
void **operator***= (const `_Expr<_Dom, _Tp>` &) const
- void `operator`*= (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **operator**+= (const `_Expr<_Dom, _Tp>` &) const
- void `operator`+= (const `valarray<_Tp>` &) const
- void `operator`-= (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **operator**-= (const `_Expr<_Dom, _Tp>` &) const
- `template<class _Dom >`
void **operator**/= (const `_Expr<_Dom, _Tp>` &) const
- void `operator`/= (const `valarray<_Tp>` &) const
- void `operator`<<= (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **operator**<<= (const `_Expr<_Dom, _Tp>` &) const
- void `operator`= (const `_Tp` &) const
- `indirect_array` & `operator`= (const `indirect_array` &)
- `template<class _Dom >`
void **operator**= (const `_Expr<_Dom, _Tp>` &) const
- void `operator`= (const `valarray<_Tp>` &) const

- `template<class _Dom >`
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator|= (const _Expr< _Dom, _Tp > &) const`

Friends

- `class gslice_array< _Tp >`
- `class valarray< _Tp >`

5.495.1 Detailed Description

`template<class _Tp> class std::indirect_array< _Tp >`

Reference to arbitrary subset of an array. An [indirect_array](#) is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an [indirect_array](#) is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned [indirect_array](#) then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an [indirect_array](#) is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

Parameters

Tp Element type.

Definition at line 61 of file `indirect_array.h`.

5.495.2 Member Function Documentation

5.495.2.1 `template<class _Tp> void std::indirect_array< _Tp >::operator%=
(const valarray< _Tp > &) const`

Modulo slice elements by corresponding elements of *v*.

5.495.2.2 `template<class _Tp> void std::indirect_array< _Tp >::operator&= (const valarray< _Tp > &) const`

Logical and slice elements with corresponding elements of *v*.

5.495.2.3 `template<class _Tp> void std::indirect_array< _Tp >::operator*= (const valarray< _Tp > &) const`

Multiply slice elements by corresponding elements of *v*.

5.495.2.4 `template<class _Tp> void std::indirect_array< _Tp >::operator+= (const valarray< _Tp > &) const`

Add corresponding elements of *v* to slice elements.

5.495.2.5 `template<class _Tp> void std::indirect_array< _Tp >::operator-= (const valarray< _Tp > &) const`

Subtract corresponding elements of *v* from slice elements.

5.495.2.6 `template<class _Tp> void std::indirect_array< _Tp >::operator/= (const valarray< _Tp > &) const`

Divide slice elements by corresponding elements of *v*.

5.495.2.7 `template<class _Tp> void std::indirect_array< _Tp >::operator<<= (const valarray< _Tp > &) const`

Left shift slice elements by corresponding elements of *v*.

5.495.2.8 `template<class _Tp> void std::indirect_array< _Tp >::operator>>= (const valarray< _Tp > &) const`

Right shift slice elements by corresponding elements of *v*.

5.495.2.9 `template<class _Tp> void std::indirect_array< _Tp >::operator^= (const valarray< _Tp > &) const`

Logical xor slice elements with corresponding elements of *v*.

5.495.2.10 `template<class _Tp> void std::indirect_array<_Tp>::operator|=
(const valarray<_Tp> &) const`

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following file:

- [indirect_array.h](#)

5.496 `std::initializer_list<_E>` Class Template Reference

[initializer_list](#)

Public Types

- `typedef const _E * const_iterator`
- `typedef const _E & const_reference`
- `typedef const _E * iterator`
- `typedef const _E & reference`
- `typedef size_t size_type`
- `typedef _E value_type`

Public Member Functions

- `const_iterator begin () const`
- `const_iterator end () const`
- `size_type size () const`

5.496.1 Detailed Description

`template<class _E> class std::initializer_list<_E>`

[initializer_list](#)

Definition at line 45 of file `initializer_list`.

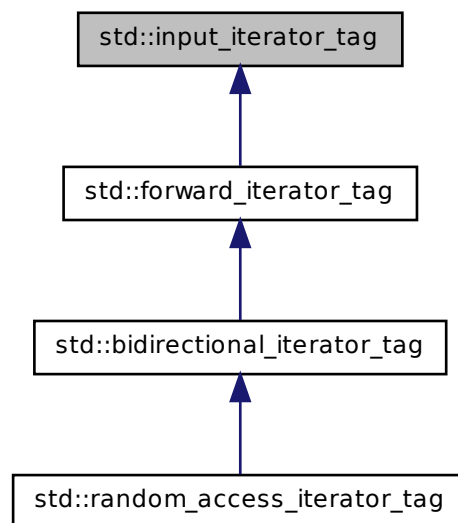
The documentation for this class was generated from the following file:

- [initializer_list](#)

5.497 std::input_iterator_tag Struct Reference

Marking input iterators.

Inheritance diagram for std::input_iterator_tag:



5.497.1 Detailed Description

Marking input iterators.

Definition at line 78 of file `stl_iterator_base_types.h`.

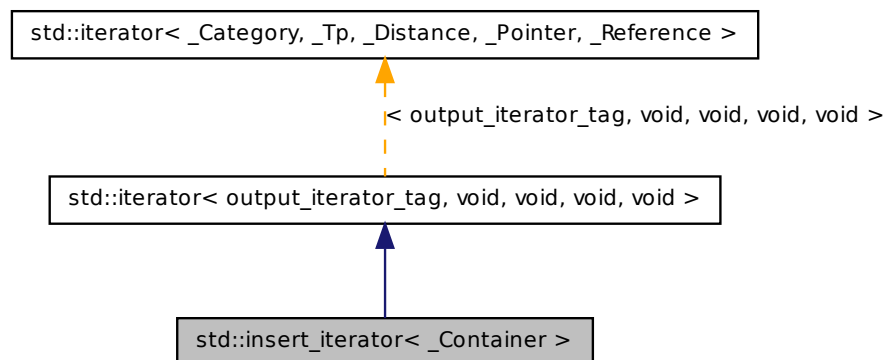
The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.498 `std::insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::insert_iterator< _Container >`:



Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator & operator*` ()
- `insert_iterator & operator++` ()
- `insert_iterator & operator++` (int)
- `insert_iterator & operator=` (`typename _Container::const_reference __value`)
- `insert_iterator & operator=` (`typename _Container::value_type &&__value`)

Protected Attributes

- `_Container * container`
- `_Container::iterator iter`

5.498.1 Detailed Description

template<typename _Container> class std::insert_iterator< _Container >

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 565 of file stl_iterator.h.

5.498.2 Member Typedef Documentation

5.498.2.1 **template<typename _Container> typedef _Container
std::insert_iterator< _Container >::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 574 of file stl_iterator.h.

5.498.2.2 **typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::difference_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

5.498.2.3 **typedef output_iterator_tag std::iterator< output_iterator_tag , void
, void , void , void >::iterator_category [inherited]**

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

5.498.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 116 of file stl_iterator_base_types.h.

5.498.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.498.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file stl_iterator_base_types.h.

5.498.3 Constructor & Destructor Documentation

5.498.3.1 `template<typename _Container> std::insert_iterator< _Container >::insert_iterator (_Container & __x, typename _Container::iterator __i) [inline]`

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 580 of file stl_iterator.h.

5.498.4 Member Function Documentation

5.498.4.1 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator* () [inline]`

Simply returns *this.

Definition at line 626 of file stl_iterator.h.

5.498.4.2 `template<typename _Container> insert_iterator&
std::insert_iterator< _Container >::operator++ (int) [inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 636 of file stl_iterator.h.

5.498.4.3 `template<typename _Container> insert_iterator&
std::insert_iterator< _Container >::operator++ () [inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 631 of file stl_iterator.h.

5.498.4.4 `template<typename _Container> insert_iterator&
std::insert_iterator< _Container >::operator= (typename
_Container::const_reference __value) [inline]`

Parameters

value An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z  
  
insert_iterator i (v, ++v.begin());  
i = 1;  
i = 2;  
i = 3;  
  
// vector v contains A, 1, 2, 3, and Z
```

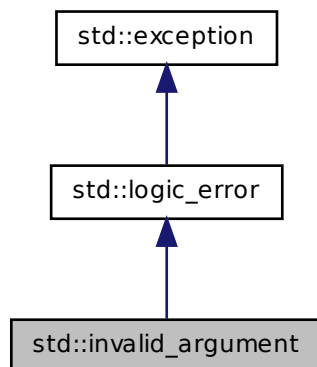
Definition at line 607 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

5.499 std::invalid_argument Class Reference

Inheritance diagram for std::invalid_argument:



Public Member Functions

- **invalid_argument** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.499.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 80 of file stdexcept.

5.499.2 Member Function Documentation

5.499.2.1 virtual const char* std::logic_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

The documentation for this class was generated from the following file:

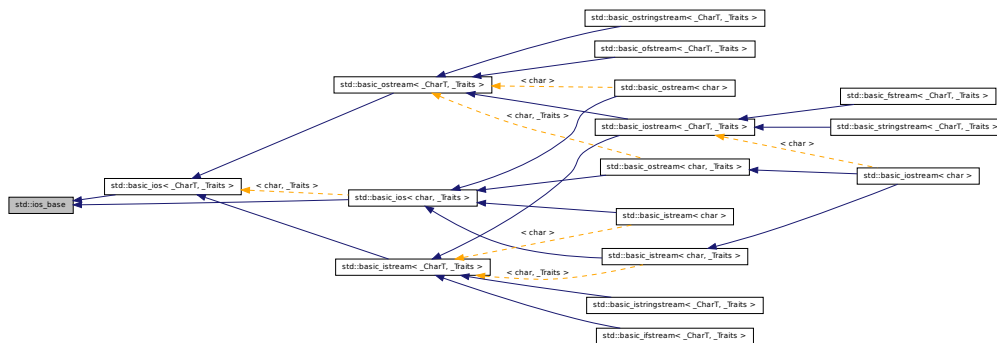
- [stdexcept](#)

5.500 std::ios_base Class Reference

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Inheritance diagram for `std::ios_base`:



Classes

- class [failure](#)

These are thrown to indicate problems with io.
 27.4.2.1.1 Class `ios_base::failure`.

Public Types

- enum [event](#) { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(* [event_callback](#))(event, `ios_base` &, int)
- typedef `_Ios_Fmtflags` [fmtflags](#)
- typedef int `io_state`

- typedef `_Ios_Iostate` [iostate](#)
- typedef int `open_mode`
- typedef `_Ios_Openmode` [openmode](#)
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` [seekdir](#)
- typedef [std::streamoff](#) `streamoff`
- typedef [std::streampos](#) `streampos`

Public Member Functions

- virtual `~ios_base` ()
- const `locale` & `_M_getloc` () const
- [fmtflags](#) `flags` () const
- [fmtflags](#) `flags` ([fmtflags](#) __fmtfl)
- `locale` `getloc` () const
- `locale` `imbue` (const `locale` & __loc) throw ()
- long & `word` (int __ix)
- [streamsize](#) `precision` ([streamsize](#) __prec)
- [streamsize](#) `precision` () const
- void *& `pword` (int __ix)
- void `register_callback` ([event_callback](#) __fn, int __index)
- [fmtflags](#) `setf` ([fmtflags](#) __fmtfl)
- [fmtflags](#) `setf` ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- void `unsetf` ([fmtflags](#) __mask)
- [streamsize](#) `width` () const
- [streamsize](#) `width` ([streamsize](#) __wide)

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) `adjustfield`
- static const [openmode](#) `app`
- static const [openmode](#) `ate`
- static const [iostate](#) `badbit`
- static const [fmtflags](#) `basefield`
- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`

- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_call_callbacks](#) ([event __ev](#)) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words & _M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()

Protected Attributes

- `_Callback_list * _M_callbacks`
- `ios_base::iostate _M_exception`
- `ios_base::fmtflags _M_flags`
- `ios_base::locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `ios_base::streamsize _M_precision`
- `ios_base::iostate _M_streambuf_state`
- `ios_base::streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

5.500.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 207 of file `ios_base.h`.

5.500.2 Member Typedef Documentation

5.500.2.1 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

5.500.2.2 typedef _Ios_Fmtflags std::ios_base::fmtflags

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

5.500.2.3 `typedef _Ios_Iostate std::ios_base::iostate`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

5.500.2.4 `typedef _Ios_Openmode std::ios_base::openmode`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

5.500.2.5 `typedef _Ios_Seekdir std::ios_base::seekdir`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.

- end, equivalent to SEEK_END in the C standard library.

Definition at line 401 of file ios_base.h.

5.500.3 Member Enumeration Documentation

5.500.3.1 enum std::ios_base::event

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 427 of file ios_base.h.

5.500.4 Constructor & Destructor Documentation

5.500.4.1 virtual std::ios_base::~ios_base () [virtual]

Invokes each callback with erase_event. Destroys local storage.

Note that the ios_base object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with erase_event (unless copyfmt is used).

5.500.5 Member Function Documentation

5.500.5.1 const locale& std::ios_base::_M_getloc () const [inline]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by std::money_get<_CharT, _InIter>::do_get(), std::num_get<_CharT, _InIter>::do_get(), std::time_get<_CharT, _InIter>::do_get_date(), std::time_get<_CharT, _InIter>::do_get_monthname(), std::time_get<_CharT, _InIter>::do_get_time(), std::time_get<_CharT, _InIter>::do_get_weekday(), std::time_get<_CharT, _InIter>::do_get_year(), std::time_put<_CharT, _OutIter>::do_put(), std::num_put<_CharT, _OutIter>::do_put(), and std::time_put<_CharT, _OutIter>::put().

5.500.5.2 fmtflags std::ios_base::flags () const [inline]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.500.5.3 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline]

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.500.5.4 locale std::ios_base::getloc () const [inline]

Locale access.

Returns

A copy of the current locale.

If imbue(loc) has previously been called, then this function returns loc. Otherwise, it returns a copy of std::locale(), the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Traits >::imbue(), std::operator>>(), and std::ws().

5.500.5.5 locale std::ios_base::imbue (const locale & __loc) throw ()

Setting a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue_event.

Reimplemented in [std::basic_ios< _CharT, _Traits >](#), and [std::basic_ios< char, _Traits >](#).

5.500.5.6 long& std::ios_base::iword (int __ix) [inline]

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios_base.h.

5.500.5.7 streamsize std::ios_base::precision () const [inline]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), and std::operator<<().

5.500.5.8 streamsize std::ios_base::precision (streamsize __prec) [inline]

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.500.5.9 void*& std::ios_base::pword (int __ix) [inline]

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.500.5.10 void std::ios_base::register_callback (event_callback __fn, int __index)

Add the callback __fn with parameter __index.

Parameters

- __fn** The function to add.
__index The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.500.5.11 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask)
[inline]**

Setting new format flags.

Parameters

- fmtfl** Additional flags to set.
mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.500.5.12 fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline]

Setting new format flags.

Parameters

- fmtfl** Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.500.5.13 `static bool std::ios_base::sync_with_stdio (bool __sync = true) [static]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.500.5.14 `void std::ios_base::unsetf (fmtflags __mask) [inline]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.500.5.15 `streamsize std::ios_base::width () const [inline]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.500.5.16 streamsize std::ios_base::width (streamsize __*wide*) [inline]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file ios_base.h.

5.500.5.17 static int std::ios_base::xalloc () throw () [static]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.500.6 Member Data Documentation**5.500.6.1** const fmtflags std::ios_base::adjustfield [static]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file ios_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.500.6.2 const openmode std::ios_base::app [static]

Seek to end before each write.

Definition at line 372 of file ios_base.h.

5.500.6.3 const openmode std::ios_base::ate [static]

Open and seek to end immediately after opening.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.500.6.4 const iostate std::ios_base::badbit [static]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.500.6.5 const fmtflags std::ios_base::basefield [static]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.500.6.6 const seekdir std::ios_base::beg [static]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios_base.h.

5.500.6.7 const openmode std::ios_base::binary [static]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.500.6.8 const fmtflags std::ios_base::boolalpha [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.500.6.9 const seekdir std::ios_base::cur [static]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.500.6.10 const fmtflags std::ios_base::dec [static]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios_base.h.

5.500.6.11 const seekdir std::ios_base::end [static]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.500.6.12 const iostate std::ios_base::eofbit [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.500.6.13 **const iostate std::ios_base::failbit** **[static]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.500.6.14 **const fmtflags std::ios_base::fixed** **[static]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios_base.h.

5.500.6.15 **const fmtflags std::ios_base::floatfield** **[static]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file ios_base.h.

5.500.6.16 **const iostate std::ios_base::goodbit** **[static]**

Indicates all is well.

Definition at line 353 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.500.6.17 const fmtflags std::ios_base::hex [static]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.500.6.18 const openmode std::ios_base::in [static]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.500.6.19 const fmtflags std::ios_base::internal [static]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios_base.h.

5.500.6.20 const fmtflags std::ios_base::left [static]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.500.6.21 `const fmtflags std::ios_base::oct [static]`

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.500.6.22 `const openmode std::ios_base::out [static]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.500.6.23 `const fmtflags std::ios_base::right [static]`

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file `ios_base.h`.

5.500.6.24 `const fmtflags std::ios_base::scientific [static]`

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

5.500.6.25 `const fmtflags std::ios_base::showbase [static]`

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file `ios_base.h`.

5.500.6.26 `const fmtflags std::ios_base::showpoint [static]`

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file `ios_base.h`.

5.500.6.27 `const fmtflags std::ios_base::showpos [static]`

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

5.500.6.28 `const fmtflags std::ios_base::skipws` `[static]`

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

5.500.6.29 `const openmode std::ios_base::trunc` `[static]`

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

5.500.6.30 `const fmtflags std::ios_base::unitbuf` `[static]`

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

5.500.6.31 `const fmtflags std::ios_base::uppercase` `[static]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

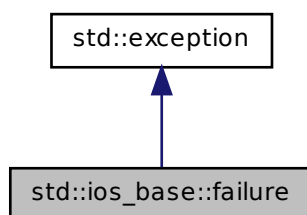
- [ios_base.h](#)

5.501 `std::ios_base::failure` Class Reference

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios_base::failure](#).

Inheritance diagram for `std::ios_base::failure`:



Public Member Functions

- **failure** (const [string](#) &__str) throw ()
- virtual const char * **what** () const throw ()

5.501.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios_base::failure](#).

Definition at line 217 of file `ios_base.h`.

5.501.2 Member Function Documentation

5.501.2.1 virtual const char* std::ios_base::failure::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [ios_base.h](#)

5.502 `std::is_base_of< _Base, _Derived >` Struct Template Reference

[is_base_of](#)

Inherits `integral_constant< bool, __is_base_of(_Base, _Derived)>`.

5.502.1 Detailed Description

```
template<typename _Base, typename _Derived> struct std::is_base_of< _Base,
_Derived >
```

[is_base_of](#)

Definition at line 305 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.503 `std::is_bind_expression< _Tp >` Struct Template Reference

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Inherits `false_type`.

5.503.1 Detailed Description

```
template<typename _Tp> struct std::is_bind_expression< _Tp >
```

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Definition at line 780 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.504 `std::is_bind_expression< _Bind< _Signature > >` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inherits `true_type`.

5.504.1 Detailed Description

```
template<typename _Signature> struct std::is_bind_expression< _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

Definition at line 1333 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.505 `std::is_bind_expression< _Bind_result< _Result, _Signature > >` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inherits `true_type`.

5.505.1 Detailed Description

```
template<typename _Result, typename _Signature> struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind` is always a bind expression.

Definition at line 1341 of file `functional`.

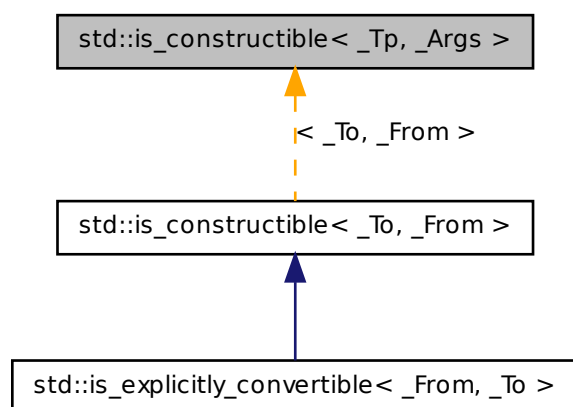
The documentation for this struct was generated from the following file:

- [functional](#)

5.506 std::is_constructible< _Tp, _Args > Struct Template Reference

[is_constructible](#)

Inheritance diagram for std::is_constructible< _Tp, _Args >:



5.506.1 Detailed Description

```
template<typename _Tp, typename... _Args> struct std::is_constructible< _Tp,  
_Args >
```

[is_constructible](#)

Definition at line 231 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.507 `std::is_convertible< _From, _To >` Struct Template Reference

[is_convertible](#)

Inherits `integral_constant< bool, __is_convertible_helper< _From, _To >::__value >`.

5.507.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_convertible< _From,
_To >
```

[is_convertible](#)

Definition at line 337 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.508 `std::is_error_code_enum< _Tp >` Struct Template Reference

[is_error_code_enum](#)

Inherits `false_type`.

5.508.1 Detailed Description

```
template<typename _Tp> struct std::is_error_code_enum< _Tp >
```

[is_error_code_enum](#)

Definition at line 52 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system_error](#)

5.509 `std::is_error_condition_enum< _Tp >` Struct Template Reference

[`is_error_condition_enum`](#)

Inherits `false_type`.

5.509.1 Detailed Description

```
template<typename _Tp> struct std::is_error_condition_enum< _Tp >
```

[`is_error_condition_enum`](#)

Definition at line 56 of file `system_error`.

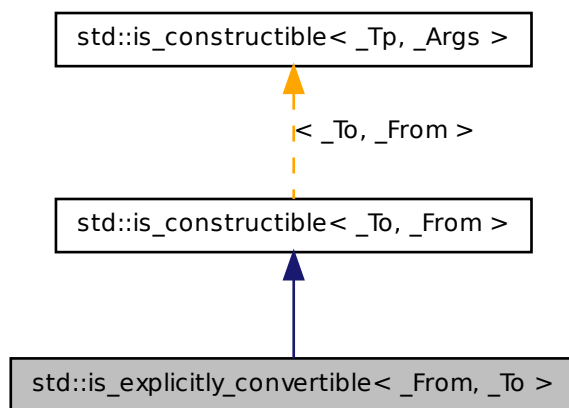
The documentation for this struct was generated from the following file:

- [`system_error`](#)

5.510 `std::is_explicitly_convertible< _From, _To >` Struct Template Reference

[`is_explicitly_convertible`](#)

Inheritance diagram for `std::is_explicitly_convertible<_From, _To>`:



5.510.1 Detailed Description

```
template<typename _From,   typename _To> struct std::is_explicitly_convertible<_From, _To>
```

[is_explicitly_convertible](#)

Definition at line 344 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.511 `std::is_lvalue_reference< typename >` Struct Template Reference

[is_lvalue_reference](#)

Inherits `false_type`.

5.512 `std::is_nothrow_constructible< _Tp, _Args >` Struct Template Reference 2577

5.511.1 Detailed Description

`template<typename> struct std::is_lvalue_reference< typename >`

[is_lvalue_reference](#)

Definition at line 69 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.512 `std::is_nothrow_constructible< _Tp, _Args >` Struct Template Reference

[is_nothrow_constructible](#)

Inherits `integral_constant< bool, __is_nt_constructible_helper< is_constructible< _Tp, _Args...>::value, _Tp, _Args...>::__value >`.

5.512.1 Detailed Description

`template<typename _Tp, typename... _Args> struct std::is_nothrow_constructible< _Tp, _Args >`

[is_nothrow_constructible](#)

Definition at line 253 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.513 `std::is_placeholder< _Tp >` Struct Template Reference

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Inherits `integral_constant< int, 0 >`.

5.513.1 Detailed Description

template<typename _Tp> struct std::is_placeholder< _Tp >

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Definition at line 789 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.514 `std::is_placeholder< _Placeholder< _Num > >` Struct Template Reference

Inherits `integral_constant< int, _Num >`.

5.514.1 Detailed Description

template<int _Num> struct std::is_placeholder< _Placeholder< _Num > >

Partial specialization of [is_placeholder](#) that provides the placeholder number for the placeholder objects defined by `libstdc++`.

Definition at line 846 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.515 `std::is_pod< _Tp >` Struct Template Reference

[is_pod](#)

Inherits `integral_constant< bool, __is_pod(_Tp)>`.

5.515.1 Detailed Description

template<typename _Tp> struct std::is_pod< _Tp >

[is_pod](#)

Definition at line 191 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.516 `std::is_reference< _Tp >` Struct Template Reference

[is_reference](#)

Inherits `integral_constant< bool,(is_lvalue_reference< _Tp >::value||is_rvalue_reference< _Tp >::value)>`.

5.516.1 Detailed Description

```
template<typename _Tp> struct std::is_reference< _Tp >
```

[is_reference](#)

Definition at line 89 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.517 `std::is_rvalue_reference< typename >` Struct Template Reference

[is_rvalue_reference](#)

Inherits `false_type`.

5.517.1 Detailed Description

```
template<typename> struct std::is_rvalue_reference< typename >
```

[is_rvalue_reference](#)

Definition at line 78 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.518 `std::is_signed< _Tp >` Struct Template Reference

[is_signed](#)

Inherits `integral_constant< bool, __is_signed_helper< _Tp >::value >`.

5.518.1 Detailed Description

```
template<typename _Tp> struct std::is_signed< _Tp >
```

[is_signed](#)

Definition at line 163 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.519 `std::is_standard_layout< _Tp >` Struct Template Reference

[is_standard_layout](#)

Inherits `integral_constant< bool, __is_standard_layout(_Tp)>`.

5.519.1 Detailed Description

```
template<typename _Tp> struct std::is_standard_layout< _Tp >
```

[is_standard_layout](#)

Definition at line 184 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.520 `std::is_trivial< _Tp >` Struct Template Reference

[is_trivial](#)

Inherits `integral_constant< bool, __is_trivial(_Tp)>`.

5.520.1 Detailed Description

```
template<typename _Tp> struct std::is_trivial< _Tp >
```

[is_trivial](#)

Definition at line 178 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.521 `std::is_unsigned< _Tp >` Struct Template Reference

[is_unsigned](#)

Inherits `integral_constant< bool,(is_arithmetic< _Tp >::value &&!is_signed< _Tp >::value)>`.

5.521.1 Detailed Description

```
template<typename _Tp> struct std::is_unsigned< _Tp >
```

[is_unsigned](#)

Definition at line 169 of file `type_traits`.

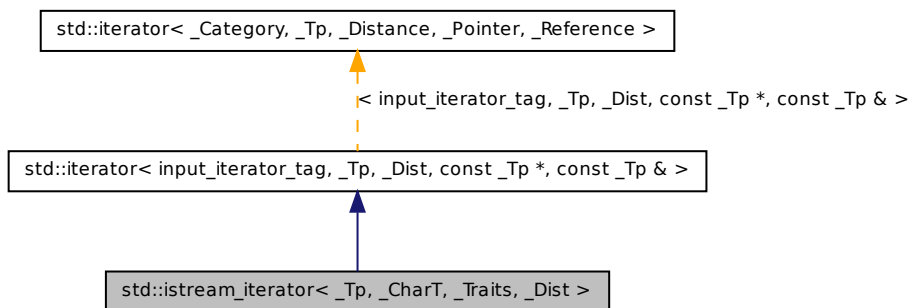
The documentation for this struct was generated from the following file:

- [type_traits](#)

5.522 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference

Provides input iterator semantics for streams.

Inheritance diagram for `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Dist` **difference_type**
- typedef `basic_istream< _CharT, _Traits >` **istream_type**
- typedef `input_iterator_tag` **iterator_category**
- typedef `const _Tp *` **pointer**
- typedef `const _Tp &` **reference**
- typedef `_Traits` **traits_type**
- typedef `_Tp` **value_type**

Public Member Functions

- `istream_iterator()`
- `istream_iterator(istream_type &__s)`
- `istream_iterator(const istream_iterator &__obj)`
- `bool _M_equal(const istream_iterator &__x) const`
- `const _Tp & operator*() const`
- `istream_iterator & operator++()`
- `istream_iterator operator++(int)`
- `const _Tp * operator->() const`

5.522.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
```

Provides input iterator semantics for streams.

Definition at line 47 of file stream_iterator.h.

5.522.2 Member Typedef Documentation

5.522.2.1 `typedef _Dist std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

5.522.2.2 `typedef input_iterator_tag std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

5.522.2.3 `typedef const _Tp * std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 116 of file stl_iterator_base_types.h.

5.522.2.4 `typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.522.2.5 `typedef _Tp std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file `std_iterator_base_types.h`.

5.522.3 Constructor & Destructor Documentation

5.522.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator () [inline]`

Construct end of input stream iterator.

Definition at line 62 of file `stream_iterator.h`.

5.522.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (istream_type & __s) [inline]`

Construct start of input stream iterator.

Definition at line 66 of file `stream_iterator.h`.

The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

5.523 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

Provides input iterator semantics for streambufs.

Inheritance diagram for `std::istreambuf_iterator< _CharT, _Traits >`:



Public Types

- `typedef _Traits::off_type` [difference_type](#)
- `typedef` [input_iterator_tag](#) [iterator_category](#)
- `typedef _CharT *` [pointer](#)

5.523 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference 2585

- typedef _CharT & [reference](#)
- typedef _CharT [value_type](#)
- typedef _CharT [char_type](#)
- typedef _Traits [traits_type](#)
- typedef _Traits::int_type [int_type](#)
- typedef [basic_streambuf](#)< _CharT, _Traits > [streambuf_type](#)
- typedef [basic_istream](#)< _CharT, _Traits > [istream_type](#)

Public Member Functions

- [istreambuf_iterator](#) () throw ()
- [istreambuf_iterator](#) ([istream_type](#) &__s) throw ()
- [istreambuf_iterator](#) ([streambuf_type](#) *__s) throw ()
- bool [equal](#) (const [istreambuf_iterator](#) &__b) const
- [char_type](#) [operator*](#) () const
- [istreambuf_iterator](#) [operator++](#) (int)
- [istreambuf_iterator](#) & [operator++](#) ()

Friends

- template<bool _IsMove, typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__
type __[copy_move_a2](#) ([istreambuf_iterator](#)< _CharT2 >, [istreambuf_iterator](#)<
_CharT2 >, _CharT2 *)
- template<typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, [ostreambuf_
iterator](#)< _CharT2 > >::__type [copy](#) ([istreambuf_iterator](#)< _CharT2 >,
[istreambuf_iterator](#)< _CharT2 >, [ostreambuf_iterator](#)< _CharT2 >)
- template<typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, [istreambuf_
iterator](#)< _CharT2 > >::__type [find](#) ([istreambuf_iterator](#)< _CharT2 >,
[istreambuf_iterator](#)< _CharT2 >, const _CharT2 &)

5.523.1 Detailed Description

**template<typename _CharT, typename _Traits> class std::istreambuf_iterator<
_CharT, _Traits >**

Provides input iterator semantics for streambufs.

Definition at line 50 of file streambuf_iterator.h.

5.523.2 Member Typedef Documentation

5.523.2.1 `template<typename _CharT , typename _Traits > typedef _CharT
std::istreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 58 of file streambuf_iterator.h.

5.523.2.2 `typedef _Traits::off_type std::iterator< input_iterator_tag , _CharT
, _Traits::off_type , _CharT * , _CharT & >::difference_type
[inherited]`

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

5.523.2.3 `template<typename _CharT , typename _Traits > typedef
_Traits::int_type std::istreambuf_iterator< _CharT, _Traits
>::int_type`

Public typedefs.

Definition at line 60 of file streambuf_iterator.h.

5.523.2.4 `template<typename _CharT , typename _Traits > typedef
basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT,
_Traits >::istream_type`

Public typedefs.

Definition at line 62 of file streambuf_iterator.h.

5.523.2.5 `typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT
, _Traits::off_type , _CharT * , _CharT & >::iterator_category
[inherited]`

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

5.523.2.6 `typedef _CharT * std::iterator< input_iterator_tag , _CharT ,
_Traits::off_type , _CharT * , _CharT & >::pointer [inherited]`

This type represents a pointer-to-value_type.

5.523 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

Definition at line 116 of file stl_iterator_base_types.h.

5.523.2.7 `typedef _CharT & std::iterator< input_iterator_tag , _CharT
, _Traits::off_type , _CharT * , _CharT & >::reference
[inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.523.2.8 `template<typename _CharT , typename _Traits > typedef
basic_streambuf< _CharT, _Traits> std::istreambuf_iterator<
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 61 of file streambuf_iterator.h.

5.523.2.9 `template<typename _CharT , typename _Traits > typedef _Traits
std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 59 of file streambuf_iterator.h.

5.523.2.10 `typedef _CharT std::iterator< input_iterator_tag , _CharT
, _Traits::off_type , _CharT * , _CharT & >::value_type
[inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file stl_iterator_base_types.h.

5.523.3 Constructor & Destructor Documentation

5.523.3.1 `template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
) throw () [inline]`

Construct end of input stream iterator.

Definition at line 96 of file streambuf_iterator.h.

5.523.3.2 `template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
istream_type & __s) throw () [inline]`

Construct start of input stream iterator.

Definition at line 100 of file streambuf_iterator.h.

5.523.3.3 `template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
streambuf_type * __s) throw () [inline]`

Construct start of streambuf iterator.

Definition at line 104 of file streambuf_iterator.h.

5.523.4 Member Function Documentation

5.523.4.1 `template<typename _CharT , typename _Traits > bool
std::istreambuf_iterator< _CharT, _Traits >::equal (const
istreambuf_iterator< _CharT, _Traits > & __b) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 160 of file streambuf_iterator.h.

5.523.4.2 `template<typename _CharT , typename _Traits > char_type
std::istreambuf_iterator< _CharT, _Traits >::operator* () const
[inline]`

Return the current character pointed to by iterator. This returns `/// streambuf.sgetc()`. It cannot be assigned. NB: The result of `/// operator*()` on an end of stream is undefined.

Definition at line 111 of file streambuf_iterator.h.

5.523.4.3 `template<typename _CharT , typename _Traits >
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits
>::operator++ (int) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 140 of file streambuf_iterator.h.

References `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

5.524 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference 2589

5.523.4.4 `template<typename _CharT, typename _Traits >
istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits
>::operator++() [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 125 of file `streambuf_iterator.h`.

References `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

The documentation for this class was generated from the following file:

- [streambuf_iterator.h](#)

5.524 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference

Common iterator class.

Inheritance diagram for `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`:



Public Types

- typedef `_Distance` [difference_type](#)
- typedef `_Category` [iterator_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value_type](#)

5.524.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t,
typename _Pointer = _Tp*, typename _Reference = _Tp&> struct std::iterator<
_Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class. This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 107 of file `stl_iterator_base_types.h`.

5.524.2 Member Typedef Documentation

5.524.2.1 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type`

Distance between iterators is represented as this type.

Reimplemented in [std::reverse_iterator< _Iterator >](#).

Definition at line 114 of file `stl_iterator_base_types.h`.

5.524.2.2 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category`

One of the [tag types](#).

Definition at line 110 of file `stl_iterator_base_types.h`.

5.524.2.3 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-value_type.

Reimplemented in [std::reverse_iterator< _Iterator >](#).

Definition at line 116 of file `stl_iterator_base_types.h`.

5.524.2.4 `template<typename _Category, typename _Tp, typename _Distance
= ptrdiff_t, typename _Pointer = _Tp*, typename _Reference =
_Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance,
_Pointer, _Reference >::reference`

This type represents a reference-to-value_type.

Reimplemented in [std::reverse_iterator< _Iterator >](#).

Definition at line 118 of file `stl_iterator_base_types.h`.

5.524.2.5 `template<typename _Category, typename _Tp, typename _Distance
= ptrdiff_t, typename _Pointer = _Tp*, typename _Reference
= _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance,
_Pointer, _Reference >::value_type`

The type "pointed to" by the iterator.

Definition at line 112 of file `stl_iterator_base_types.h`.

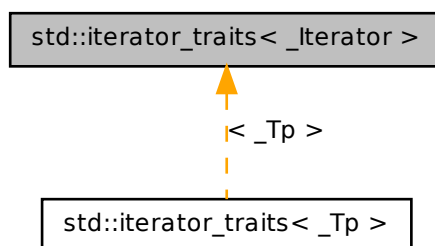
The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.525 `std::iterator_traits< _Iterator >` Struct Template Reference

Traits class for iterators.

Inheritance diagram for `std::iterator_traits<_Iterator>`:



Public Types

- `typedef _Iterator::difference_type` **difference_type**
- `typedef _Iterator::iterator_category` **iterator_category**
- `typedef _Iterator::pointer` **pointer**
- `typedef _Iterator::reference` **reference**
- `typedef _Iterator::value_type` **value_type**

5.525.1 Detailed Description

template<typename _Iterator> struct std::iterator_traits<_Iterator>

Traits class for iterators. This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the `Iterator` argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

Definition at line 130 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.526 `std::iterator_traits< _Tp * >` Struct Template Reference

Partial specialization for pointer types.

Public Types

- typedef ptrdiff_t **difference_type**
- typedef [random_access_iterator_tag](#) **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

5.526.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits< _Tp * >
```

Partial specialization for pointer types.

Definition at line 141 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.527 `std::iterator_traits< const _Tp * >` Struct Template Reference

Partial specialization for const pointer types.

Public Types

- typedef ptrdiff_t **difference_type**
- typedef [random_access_iterator_tag](#) **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

5.527.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

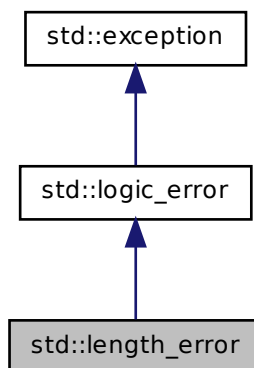
Definition at line 152 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.528 `std::length_error` Class Reference

Inheritance diagram for `std::length_error`:



Public Member Functions

- **`length_error`** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.528.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a [basic_string](#) instance).

Definition at line 88 of file `stdexcept`.

5.528.2 Member Function Documentation

5.528.2.1 `virtual const char* std::logic_error::what () const throw ()`
[[virtual](#), [inherited](#)]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

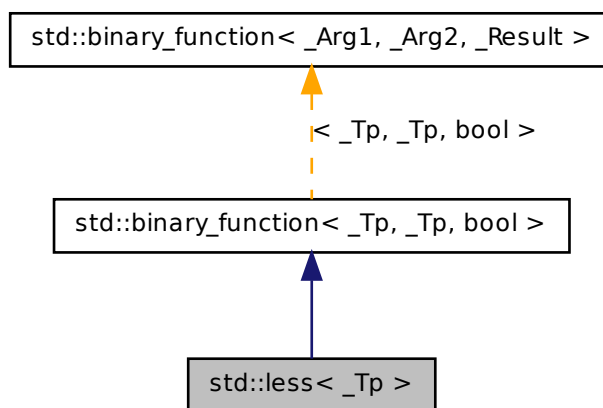
The documentation for this class was generated from the following file:

- [stdexcept](#)

5.529 `std::less<_Tp>` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::less< _Tp >`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.529.1 Detailed Description

`template<typename _Tp> struct std::less< _Tp >`

One of the [comparison functors](#).

Definition at line 226 of file `stl_function.h`.

5.529.2 Member Typedef Documentation

5.529.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.529.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.529.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

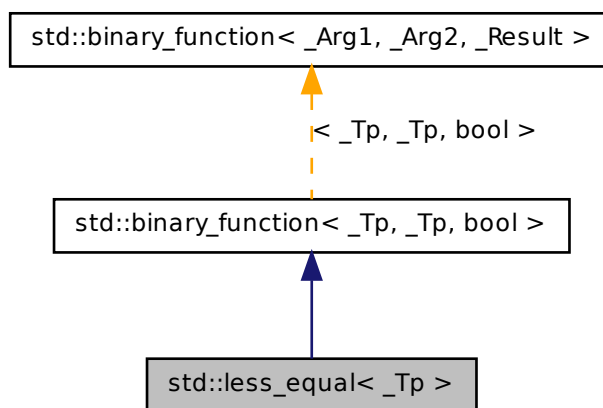
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.530 std::less_equal< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::less_equal<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.530.1 Detailed Description

`template<typename _Tp> struct std::less_equal<_Tp>`

One of the [comparison functors](#).

Definition at line 244 of file `stl_function.h`.

5.530.2 Member Typedef Documentation

5.530.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.530.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.530.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.531 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference

A model of a linear congruential random number generator.

Public Types

- `typedef _UIntType result_type`

Public Member Functions

- `linear_congruential_engine (result_type __s=default_seed)`
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_-
congruential_engine>::value> ::type>
linear_congruential_engine (_Sseq &__q)`

- void [discard](#) (unsigned long long __z)
- [result_type](#) max () const
- [result_type](#) min () const
- [result_type](#) operator() ()
- void [seed](#) ([result_type](#) __s=default_seed)
- template<typename _Sseq >
[std::enable_if](#)< std::is_class< _Sseq >::value >::type [seed](#) (_Sseq &__q)

Static Public Attributes

- static const [result_type](#) **default_seed**
- static const [result_type](#) **increment**
- static const [result_type](#) **modulus**
- static const [result_type](#) **multiplier**

Friends

- template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::linear_congruential_engine](#)< _UIntType1, __a1, __c1, __m1 > &)
- bool [operator==](#) (const [linear_congruential_engine](#) &__lhs, const [linear_congruential_engine](#) &__rhs)
- template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::linear_congruential_engine](#)< _UIntType1, __a1, __c1, __m1 > &)

5.531.1 Detailed Description

template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> class [std::linear_congruential_engine](#)< _UIntType, __a, __c, __m >

A model of a linear congruential random number generator. A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is [std::numeric_limits<_UIntType>::max\(\)](#) plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 157 of file random.h.

5.531.2 Member Typedef Documentation

5.531.2.1 `template<typename _UIntType, _UIntType __a,
_UIntType __c, _UIntType __m> typedef _UIntType
std::linear_congruential_engine< _UIntType, __a, __c, __m
>::result_type`

The type of the generated random value.

Definition at line 166 of file random.h.

5.531.3 Constructor & Destructor Documentation

5.531.3.1 `template<typename _UIntType, _UIntType __a, _UIntType __c,
_UIntType __m> std::linear_congruential_engine< _UIntType,
__a, __c, __m >::linear_congruential_engine (result_type __s =
default_seed) [inline, explicit]`

Constructs a linear_congruential_engine random number generator engine with seed __s. The default seed value is 1.

Parameters

__s The initial seed value.

Definition at line 184 of file random.h.

References std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

5.531.3.2 `template<typename _UIntType, _UIntType __a, _UIntType
__c, _UIntType __m> template<typename _Sseq, typename =
typename std::enable_if<!std::is_same<_Sseq, linear_congruential_
engine>::value> ::type> std::linear_congruential_engine<
_UIntType, __a, __c, __m >::linear_congruential_engine (_Sseq &
__q) [inline, explicit]`

Constructs a linear_congruential_engine random number generator engine seeded from the seed sequence __q.

Parameters

__q the seed sequence.

Definition at line 197 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed()`.

5.531.4 Member Function Documentation

5.531.4.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine<_UIntType, __a, __c, __m>::discard (unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 247 of file random.h.

5.531.4.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::max () const [inline]`

Gets the largest possible value in the output range.

Todo

This should be constexpr.

Definition at line 238 of file random.h.

5.531.4.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::min () const [inline]`

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be > 0 , otherwise 0 is allowed.

Todo

This should be constexpr.

Definition at line 229 of file random.h.

5.531.4.4 `template<typename _UIntType, _UIntType __a, _UIntType __c,
 _UIntType __m> result_type std::linear_congruential_engine<
 _UIntType, __a, __c, __m >::operator() () [inline]`

Gets the next random number in the sequence.

Definition at line 257 of file random.h.

5.531.4.5 `template<typename _UIntType , _UIntType __a, _UIntType
 __c, _UIntType __m> template<typename _Sseq >
 std::enable_if< std::is_class< _Sseq >::value >::type
 std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
 _Sseq & __q)`

Reseeds the linear_congruential_engine random number generator engine sequence using values from the seed sequence __q.

Parameters

__q the seed sequence.

Seeds the LCR engine with a value generated by __q.

Definition at line 142 of file random.tcc.

References std::__lg(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

5.531.4.6 `template<typename _UIntType , _UIntType __a, _UIntType __c,
 _UIntType __m> void std::linear_congruential_engine< _UIntType,
 __a, __c, __m >::seed (result_type __s = default_seed)`

Reseeds the linear_congruential_engine random number generator engine sequence to the seed __s.

Parameters

__s The new seed.

Seeds the LCR with integral value __s, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 126 of file random.tcc.

Referenced by std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

5.531.5 Friends And Related Function Documentation

5.531.5.1 `template<typename _UIntType, _UIntType __a, _UIntType __c,
_UIntType __m> template<typename _UIntType1, _UIntType1
__a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT
, typename _Traits > std::basic_ostream<_CharT, _Traits>&
operator<< (std::basic_ostream<_CharT, _Traits> &, const
std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1 > &
) [friend]`

Writes the textual representation of the state $x(i)$ of x to `__os`.

Parameters

`__os` The output stream.

`__lcr` A % [linear_congruential_engine](#) random number generator.

Returns

`__os`.

5.531.5.2 `template<typename _UIntType, _UIntType __a,
_UIntType __c, _UIntType __m> bool operator==(const
linear_congruential_engine<_UIntType, __a, __c, __m> & __lhs,
const linear_congruential_engine<_UIntType, __a, __c, __m> &
__rhs) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

Parameters

`__lhs` A linear congruential random number generator object.

`__rhs` Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 275 of file random.h.

5.531.5.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &, std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> &) [friend]`

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

Parameters

`__is` The input stream.

`__lcr` A % [linear_congruential_engine](#) random number generator.

Returns

`__is`.

5.531.6 Member Data Documentation

5.531.6.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine<_UIntType, __a, __c, __m>::increment [static]`

An increment.

Definition at line 171 of file `random.h`.

5.531.6.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine<_UIntType, __a, __c, __m>::modulus [static]`

The modulus.

Definition at line 173 of file `random.h`.

5.531.6.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine<_UIntType, __a, __c, __m>::multiplier [static]`

The multiplier.

Definition at line 169 of file random.h.

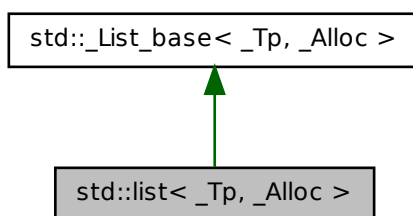
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.532 `std::list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::list< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_List_const_iterator< _Tp >` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- [list](#) ()
- [list](#) (const allocator_type &__a)
- [list](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
[list](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- [list](#) (const [list](#) &__x)
- [list](#) (size_type __n)
- [list](#) ([list](#) &&__x)
- [list](#) ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename _InputIterator >
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) ([initializer_list](#)< value_type > __l)
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- template<typename... _Args>
[iterator](#) [emplace](#) ([iterator](#) __position, _Args &&...__args)
- template<typename... _Args>
void [emplace_back](#) (_Args &&...__args)
- template<typename... _Args>
void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- iterator [erase](#) ([iterator](#) __position)
- iterator [erase](#) ([iterator](#) __first, [iterator](#) __last)
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const
- iterator [insert](#) ([iterator](#) __position, const value_type &__x)
- iterator [insert](#) ([iterator](#) __position, value_type &&__x)

- void [insert](#) (iterator __p, [initializer_list](#)< value_type > __l)
- void [insert](#) (iterator __position, size_type __n, const value_type &__x)
- template<typename _InputIterator >
void [insert](#) (iterator __position, _InputIterator __first, _InputIterator __last)
- size_type [max_size](#) () const
- void [merge](#) (list &&__x)
- void [merge](#) (list &__x)
- template<typename _StrictWeakOrdering >
void [merge](#) (list &&, _StrictWeakOrdering)
- template<typename _StrictWeakOrdering >
void [merge](#) (list &__x, _StrictWeakOrdering __comp)
- list & operator= (const list &__x)
- list & operator= (list &&__x)
- list & operator= ([initializer_list](#)< value_type > __l)
- void [pop_back](#) ()
- void [pop_front](#) ()
- void [push_back](#) (const value_type &__x)
- void [push_back](#) (value_type &&__x)
- void [push_front](#) (value_type &&__x)
- void [push_front](#) (const value_type &__x)
- [reverse_iterator](#) rbegin ()
- [const_reverse_iterator](#) rbegin () const
- void [remove](#) (const _Tp &__value)
- template<typename _Predicate >
void [remove_if](#) (_Predicate)
- [reverse_iterator](#) rend ()
- [const_reverse_iterator](#) rend () const
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const value_type &__x)
- void [reverse](#) ()
- size_type [size](#) () const
- template<typename _StrictWeakOrdering >
void [sort](#) (_StrictWeakOrdering)
- void [sort](#) ()
- void [splice](#) (iterator __position, list &__x, iterator __i)
- void [splice](#) (iterator __position, list &&__x, iterator __i)
- void [splice](#) (iterator __position, list &__x)
- void [splice](#) (iterator __position, list &__x, iterator __first, iterator __last)
- void [splice](#) (iterator __position, list &&__x, iterator __first, iterator __last)
- void [splice](#) (iterator __position, list &&__x)
- void [swap](#) (list &__x)
- template<typename _BinaryPredicate >
void [unique](#) (_BinaryPredicate)
- void [unique](#) ()

Protected Types

- typedef [_List_node](#)< _Tp > **_Node**
- typedef _Alloc::template rebind< [_List_node](#)< _Tp > >::other **_Node_alloc_type**

Protected Member Functions

- template<typename _Integer >
void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator >
void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_check_equal_allocators** ([list](#) &__x)
- void **_M_clear** ()
- template<typename... _Args>
[_Node](#) * **_M_create_node** (_Args &&...__args)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_erase** ([iterator](#) __position)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__x)
- [_List_node](#)< _Tp > * **_M_get_node** ()
- [_Node_alloc_type](#) & **_M_get_Node_allocator** ()
- const [_Node_alloc_type](#) & **_M_get_Node_allocator** () const
- [_Tp_alloc_type](#) **_M_get_Tp_allocator** () const
- void **_M_init** ()
- template<typename _InputIterator >
void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer >
void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename... _Args>
void **_M_insert** ([iterator](#) __position, _Args &&...__args)
- void **_M_put_node** ([_List_node](#)< _Tp > *__p)
- void **_M_transfer** ([iterator](#) __position, [iterator](#) __first, [iterator](#) __last)

Protected Attributes

- [_List_impl](#) **_M_impl**

5.532.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::list<_Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike [std::vector](#) and [std::deque](#), random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, [std::list](#) provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

```
A <---> B <---> C <---> D
```

is actually circular; a link exists between `A` and `D`. The list class holds (as its only data member) a private `list::iterator` pointing to `D`, not to `A`! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 417 of file `stl_list.h`.

5.532.2 Constructor & Destructor Documentation

```
5.532.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list ( ) [inline]
```

Default constructor creates no elements.

Definition at line 500 of file `stl_list.h`.

5.532.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (const allocator_type & __a)
[inline, explicit]`

Creates a list with no elements.

Parameters

a An allocator object.

Definition at line 508 of file stl_list.h.

5.532.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (size_type __n) [inline,
explicit]`

Creates a list with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor fills the list with *n* default constructed elements.

Definition at line 520 of file stl_list.h.

5.532.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (size_type __n, const value_type &
__value, const allocator_type & __a = allocator_type())
[inline]`

Creates a list with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

a An allocator object.

This constructor fills the list with *n* copies of *value*.

Definition at line 532 of file stl_list.h.

```
5.532.2.5  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             std::list<_Tp, _Alloc>::list ( const list<_Tp, _Alloc> & __x )
             [inline]
```

List copy constructor.

Parameters

x A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *x*.

Definition at line 559 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

```
5.532.2.6  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             std::list<_Tp, _Alloc>::list ( list<_Tp, _Alloc> && __x )
             [inline]
```

List move constructor.

Parameters

x A list of identical element and allocator types.

The newly-created list contains the exact contents of *x*. The contents of *x* are a valid, but unspecified list.

Definition at line 571 of file `stl_list.h`.

```
5.532.2.7  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             std::list<_Tp, _Alloc>::list ( initializer_list<value_type> & __l,
             const allocator_type & __a = allocator_type() ) [inline]
```

Builds a list from an [initializer_list](#).

Parameters

l An [initializer_list](#) of `value_type`.

a An allocator object.

Create a list consisting of copies of the elements in the [initializer_list](#) *l*. This is linear in `l.size()`.

Definition at line 582 of file `stl_list.h`.

5.532.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename _InputIterator > std::list< _Tp, _Alloc >::list (
 _InputIterator __first, _InputIterator __last, const allocator_type &
 __a = allocator_type()) [inline]`

Builds a list from a range.

Parameters

first An input iterator.

last An input iterator.

a An allocator object.

Create a list consisting of copies of the elements from *[first,last)*. This is linear in N (where N is distance(*first,last*)).

Definition at line 599 of file stl_list.h.

5.532.3 Member Function Documentation

5.532.3.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename... _Args> _Node* std::list< _Tp, _Alloc
 >::_M_create_node (_Args &&... __args) [inline,
 protected]`

Parameters

x An instance of user data.

Allocates space for a new node and constructs a copy of *x* in it.

Definition at line 477 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::emplace(), and std::list< _Tp, _Alloc >::insert().

5.532.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::list< _Tp, _Alloc >::assign (size_type __n, const
 value_type & __val) [inline]`

Assigns a given value to a list.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a list with n copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 670 of file `stl_list.h`.

```
5.532.3.3  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            template<typename _InputIterator > void std::list<_Tp, _Alloc
            >::assign ( _InputIterator __first, _InputIterator __last )
            [inline]
```

Assigns a range to a list.

Parameters

first An input iterator.

last An input iterator.

This function fills a list with copies of the elements in the range $[first, last)$.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 687 of file `stl_list.h`.

```
5.532.3.4  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            void std::list<_Tp, _Alloc >::assign ( initializer_list< value_type >
            __l ) [inline]
```

Assigns an [initializer_list](#) to a list.

Parameters

l An [initializer_list](#) of `value_type`.

Replace the contents of the list with copies of the elements in the [initializer_list](#) *l*. This is linear in `l.size()`.

Definition at line 703 of file `stl_list.h`.

References `std::list<_Tp, _Alloc >::assign()`.

Referenced by `std::list<_Tp, _Alloc >::assign()`.

```
5.532.3.5  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            reference std::list<_Tp, _Alloc >::back ( ) [inline]
```

Returns a read/write reference to the data at the last element of the list.

Definition at line 903 of file stl_list.h.

5.532.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::back () const [inline]`

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 915 of file stl_list.h.

5.532.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::begin () [inline]`

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 718 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::list(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::operator=(), std::operator==(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::splice(), and std::list< _Tp, _Alloc >::unique().

5.532.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 727 of file stl_list.h.

5.532.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 791 of file stl_list.h.

5.532.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 800 of file stl_list.h.

5.532.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::clear () [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1182 of file `stl_list.h`.

5.532.3.12 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::crbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 809 of file `stl_list.h`.

5.532.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::crend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 818 of file `stl_list.h`.

5.532.3.14 `template<typename _Tp, typename _Alloc> template<typename...
_Args> list<_Tp, _Alloc>::iterator list::emplace (iterator
__position, _Args &&... __args)`

Constructs object in list before specified iterator.

Parameters

position A const_iterator into the list.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.`

Definition at line 87 of file list.tcc.

References std::list< _Tp, _Alloc >::_M_create_node().

5.532.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty () const [inline]`

Returns true if the list is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Definition at line 828 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::sort(), and std::list< _Tp, _Alloc >::splice().

5.532.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 736 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::list(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::operator=(), std::operator==(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::splice(), and std::list< _Tp, _Alloc >::unique().

5.532.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 745 of file stl_list.h.

5.532.3.18 `template<typename _Tp , typename _Alloc > list< _Tp, _Alloc
>::iterator list::erase (iterator __position)`

Remove element at given position.

Parameters

position Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or [end\(\)](#)).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 108 of file list.tcc.

Referenced by `std::list< _Tp, _Alloc >::operator=()`, and `std::list< _Tp, _Alloc >::resize()`.

5.532.3.19 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::erase (iterator __first, iterator
__last) [inline]`

Remove a range of elements.

Parameters

first Iterator pointing to the first element to be erased.

last Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1148 of file stl_list.h.

5.532.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front () [inline]`

Returns a read/write reference to the data at the first element of the list.

Definition at line 887 of file stl_list.h.

5.532.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 895 of file stl_list.h.

5.532.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator () const
[inline]`

Get a copy of the memory allocation object.

Reimplemented from [std::List_base< _Tp, _Alloc >](#).

Definition at line 709 of file stl_list.h.

5.532.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (iterator __position,
value_type && __x) [inline]`

Inserts given rvalue into list before specified iterator.

Parameters

position An iterator into the list.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1049 of file stl_list.h.

5.532.3.24 `template<typename _Tp , typename _Alloc > list< _Tp, _Alloc
>::iterator list::insert (iterator __position, const value_type &
__x)`

Inserts given value into list before specified iterator.

Parameters

position An iterator into the list.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 98 of file list.tcc.

References `std::list< _Tp, _Alloc >::_M_create_node()`.

Referenced by `std::list< _Tp, _Alloc >::operator=()`, and `std::list< _Tp, _Alloc >::resize()`.

5.532.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::insert (iterator __p, initializer_list<
value_type > __l) [inline]`

Inserts the contents of an [initializer_list](#) into list before specified iterator.

Parameters

p An iterator into the list.

l An [initializer_list](#) of value_type.

This function will insert copies of the data in the [initializer_list l](#) into the list before the location specified by *p*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1066 of file stl_list.h.

References `std::list< _Tp, _Alloc >::insert()`.

Referenced by `std::list< _Tp, _Alloc >::insert()`.

5.532.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::insert (iterator __position, size_type
__n, const value_type & __x) [inline]`

Inserts a number of copies of given data into the list.

Parameters

position An iterator into the list.

- n* Number of elements to be inserted.
- x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1083 of file stl_list.h.

5.532.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::list< _Tp,
_Alloc >::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]`

Inserts a range into the list.

Parameters

- position* An iterator into the list.
- first* An input iterator.
- last* An input iterator.

This function will insert copies of the data in the range [*first,last*) into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1104 of file stl_list.h.

5.532.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size () const [inline]`

Returns the [size\(\)](#) of the largest possible list.

Definition at line 838 of file stl_list.h.

5.532.3.29 `template<typename _Tp , typename _Alloc > void list::merge (
list< _Tp, _Alloc > && __x)`

Merge sorted lists.

Parameters

- x* Sorted list to merge.

Assumes that both x and this list are sorted according to operator<(). Merges elements of x into this list in sorted order, leaving x empty when complete. Elements in this list precede elements in x that are equal.

Definition at line 287 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), and std::list< _Tp, _Alloc >::end().

Referenced by std::list< _Tp, _Alloc >::sort().

5.532.3.30 `template<typename _Tp, typename _Alloc > template<typename
_StrictWeakOrdering > void list::merge (list< _Tp, _Alloc > &&
__x, _StrictWeakOrdering __comp)`

Merge sorted lists according to comparison function.

Parameters

x Sorted list to merge.

StrictWeakOrdering Comparison function defining sort order.

Assumes that both x and this list are sorted according to StrictWeakOrdering. Merges elements of x into this list in sorted order, leaving x empty when complete. Elements in this list precede elements in x that are equivalent according to StrictWeakOrdering().

Definition at line 321 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), and std::list< _Tp, _Alloc >::end().

5.532.3.31 `template<typename _Tp, typename _Alloc > list< _Tp, _Alloc > &
list::operator= (const list< _Tp, _Alloc > & __x)`

List assignment operator.

No explicit dtor needed as the _Base dtor takes care of things. The _Base dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

x A list of identical element and allocator types.

All the elements of x are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 184 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::end(), std::list< _Tp, _Alloc >::erase(), and std::list< _Tp, _Alloc >::insert().

5.532.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (initializer_list<
value_type > __l) [inline]`

List initializer list assignment operator.

Parameters

l An [initializer_list](#) of value_type.

Replace the contents of the list with copies of the elements in the [initializer_list](#) *l*. This is linear in *l*.size().

Definition at line 652 of file stl_list.h.

5.532.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (list< _Tp, _Alloc > &&
__x) [inline]`

List move assignment operator.

Parameters

x A list of identical element and allocator types.

The contents of *x* are moved into this list (without copying). *x* is a valid, but unspecified list

Definition at line 635 of file stl_list.h.

5.532.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back () [inline]`

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before [pop_back\(\)](#) is called.

Definition at line 1001 of file stl_list.h.

5.532.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_front () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 961 of file `stl_list.h`.

```
5.532.3.36  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
              void std::list< _Tp, _Alloc >::push_back ( const value_type & __x
              ) [inline]
```

Add data to the end of the list.

Parameters

`x` Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 975 of file `stl_list.h`.

```
5.532.3.37  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
              void std::list< _Tp, _Alloc >::push_front ( const value_type & __x
              ) [inline]
```

Add data to the front of the list.

Parameters

`x` Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 934 of file `stl_list.h`.

```
5.532.3.38  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
              reverse_iterator std::list< _Tp, _Alloc >::rbegin ( ) [inline]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 754 of file `stl_list.h`.

5.532.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 763 of file stl_list.h.

5.532.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::remove (const _Tp & __value)`

Remove all elements equal to value.

Parameters

value The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.532.3.41 `template<typename _Tp, typename _Alloc > template<typename
_Predicate > void list::remove_if (_Predicate __pred)`

Remove all elements satisfying a predicate.

Parameters

Predicate Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 391 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), and std::list< _Tp, _Alloc >::end().

5.532.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 781 of file `std_list.h`.

5.532.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list<_Tp, _Alloc>::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 772 of file `std_list.h`.

5.532.3.44 `template<typename _Tp, typename _Alloc > void list::resize (
size_type __new_size, const value_type & __x)`

Resizes the list to the specified number of elements.

Parameters

new_size Number of elements the list should contain.

x Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 153 of file `list.tcc`.

References `std::list<_Tp, _Alloc>::begin()`, `std::list<_Tp, _Alloc>::end()`, `std::list<_Tp, _Alloc>::erase()`, and `std::list<_Tp, _Alloc>::insert()`.

5.532.3.45 `template<typename _Tp, typename _Alloc > void list::resize (
size_type __new_size)`

Resizes the list to the specified number of elements.

Parameters

new_size Number of elements the list should contain.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 138 of file `list.tcc`.

References `std::list<_Tp, _Alloc>::begin()`, `std::list<_Tp, _Alloc>::end()`, and `std::list<_Tp, _Alloc>::erase()`.

5.532.3.46 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::reverse () [inline]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1402 of file stl_list.h.

5.532.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::size () const [inline]`

Returns the number of elements in the list.

Definition at line 833 of file stl_list.h.

References `__gnu_cxx::distance()`.

5.532.3.48 `template<typename _Tp , typename _Alloc > void list::sort ()`

Sort the elements.

Sorts the elements of this list in $N\log N$ time. Equivalent elements remain in list order.

Definition at line 353 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

5.532.3.49 `template<typename _Tp , typename _Alloc > template<typename
_StrictWeakOrdering > void list::sort (_StrictWeakOrdering
__comp)`

Sort the elements according to comparison function.

Sorts the elements of this list in $N\log N$ time. Equivalent elements remain in list order.

Definition at line 430 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

```

5.532.3.50  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::list< _Tp, _Alloc >::splice ( iterator __position, list<
             _Tp, _Alloc > && __x, iterator __first, iterator __last )
             [inline]

```

Insert range from another list.

Parameters

position Iterator referencing the element to insert before.

x Source list.

first Iterator referencing the start of range in *x*.

last Iterator referencing the end of range in *x*.

Removes elements in the range [*first*,*last*) and inserts them before *position* in constant time.

Undefined if *position* is in [*first*,*last*).

Definition at line 1268 of file stl_list.h.

```

5.532.3.51  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::list< _Tp, _Alloc >::splice ( iterator __position, list<
             _Tp, _Alloc > && __x, iterator __i ) [inline]

```

Insert element from another list.

Parameters

position Iterator referencing the element to insert before.

x Source list.

i Iterator referencing the element to move.

Removes the element in list *x* referenced by *i* and inserts it into the current list before *position*.

Definition at line 1232 of file stl_list.h.

```

5.532.3.52  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::list< _Tp, _Alloc >::splice ( iterator __position, list<
             _Tp, _Alloc > && __x ) [inline]

```

Insert contents of another list.

Parameters

position Iterator referencing the element to insert before.

x Source list.

The elements of *x* are inserted in constant time in front of the element referenced by *position*. *x* becomes an empty list.

Requires this != *x*.

Definition at line 1202 of file stl_list.h.

References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::empty(), and std::list< _Tp, _Alloc >::end().

Referenced by std::list< _Tp, _Alloc >::sort().

5.532.3.53 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (list< _Tp, _Alloc > & __x)
[inline]`

Swaps data with another list.

Parameters

x A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global [std::swap\(\)](#) function is specialized such that std::swap(l1,l2) will feed to this function.

Definition at line 1165 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::sort(), and std::swap().

5.532.3.54 `template<typename _Tp , typename _Alloc > void list::unique ()`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 266 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), and std::list< _Tp, _Alloc >::end().

5.532.3.55 `template<typename _Tp , typename _Alloc > template<typename
_BinaryPredicate > void list::unique (_BinaryPredicate
__binary_pred)`

Remove consecutive elements satisfying a predicate.

Parameters

BinaryPredicate Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 409 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

The documentation for this class was generated from the following files:

- [stl_list.h](#)
- [list.tcc](#)

5.533 std::locale Class Reference

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Classes

- class [facet](#)

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class [id](#)

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

Public Types

- typedef int [category](#)

Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &__other) throw ()
- [locale](#) (const [locale](#) &__base, const char *__s, [category](#) __cat)
- [locale](#) (const [locale](#) &__base, const [locale](#) &__add, [category](#) __cat)
- [locale](#) (const char *__s)
- template<typename _Facet >
[locale](#) (const [locale](#) &__other, _Facet *__f)
- [~locale](#) () throw ()
- template<typename _Facet >
[locale combine](#) (const [locale](#) &__other) const
- [string name](#) () const
- bool [operator!=](#) (const [locale](#) &__other) const throw ()
- template<typename _Char, typename _Traits, typename _Alloc >
bool [operator\(\)](#) (const [basic_string](#)< _Char, _Traits, _Alloc > &__s1, const
[basic_string](#)< _Char, _Traits, _Alloc > &__s2) const
- template<typename _CharT, typename _Traits, typename _Alloc >
bool [operator\(\)](#) (const [basic_string](#)< _CharT, _Traits, _Alloc > &__s1, const
[basic_string](#)< _CharT, _Traits, _Alloc > &__s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &__other) throw ()
- bool [operator==](#) (const [locale](#) &__other) const throw ()

Static Public Member Functions

- static const [locale](#) & [classic](#) ()
- static [locale global](#) (const [locale](#) &)

Static Public Attributes

- static const [category none](#)
- static const [category ctype](#)
- static const [category numeric](#)
- static const [category collate](#)
- static const [category time](#)
- static const [category monetary](#)
- static const [category messages](#)
- static const [category all](#)

Friends

- struct **__use_cache**
- class **_Impl**
- class **facet**
- template<typename _Facet >
bool [has_facet](#) (const [locale](#) &) throw ()
- template<typename _Facet >
const _Facet & [use_facet](#) (const [locale](#) &)

5.533.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file locale_classes.h.

5.533.2 Member Typedef Documentation

5.533.2.1 typedef int std::locale::category

Definition of [locale::category](#).

Definition at line 67 of file locale_classes.h.

5.533.3 Constructor & Destructor Documentation

5.533.3.1 std::locale::locale () throw ()

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by [combine\(\)](#).

5.533.3.2 std::locale::locale (const locale & *__other*) throw ()

Copy constructor.

Constructs a copy of *other*.

Parameters

other The locale to copy.

5.533.3.3 std::locale::locale (const char * __s) [explicit]

Named locale constructor.

Constructs a copy of the named C library locale.

Parameters

s Name of the locale to construct.

Exceptions

[*std::runtime_error*](#) if *s* is null or an undefined locale.

5.533.3.4 std::locale::locale (const locale & __base, const char * __s, category __cat)

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

Parameters

base The locale to copy.

s Name of the locale to use facets from.

cat Set of categories defining the facets to use from *s*.

Exceptions

[*std::runtime_error*](#) if *s* is null or an undefined locale.

5.533.3.5 std::locale::locale (const locale & __base, const locale & __add, category __cat)

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

- base* The locale to copy.
- add* The locale to use facets from.
- cat* Set of categories defining the facets to use from add.

5.533.3.6 `template<typename _Facet > std::locale::locale (const locale & __other, _Facet * __f)`

Construct locale with another facet.

Constructs a copy of the locale *other*. The facet is added to , replacing an existing facet of type Facet if there is one. If is null, this locale is a copy of *other*.

Parameters

- other* The locale to copy.
- f* The facet to add in.

Definition at line 43 of file locale_classes.tcc.

5.533.3.7 `std::locale::~~locale () throw ()`

Locale destructor.

5.533.4 Member Function Documentation

5.533.4.1 `static const locale& std::locale::classic () [static]`

Return reference to the C locale.

5.533.4.2 `template<typename _Facet > locale std::locale::combine (const locale & __other) const`

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type Facet from the locale *other* into the new locale.

Parameters

- Facet* The facet type to copy from other
- other* The locale to copy from.

Returns

Newly constructed locale.

Exceptions

std::runtime_error if other has no facet of type Facet.

Definition at line 61 of file locale_classes.tcc.

References locale().

5.533.4.3 static locale std::locale::global (const locale &) [static]

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC_ALL, loc.name()).

Parameters

locale The new locale to make global.

Returns

Copy of the old global locale.

5.533.4.4 string std::locale::name () const

Return locale name.

Returns

Locale name or "*" if unnamed.

5.533.4.5 bool std::locale::operator!= (const locale & __other) const throw () [inline]

Locale inequality.

Parameters

other The locale to compare against.

Returns

! (*this == other)

Definition at line 234 of file locale_classes.h.

5.533.4.6 `template<typename _Char , typename _Traits , typename _Alloc >
bool std::locale::operator() (const basic_string< _Char, _Traits,
_Alloc > & __s1, const basic_string< _Char, _Traits, _Alloc > &
__s2) const`

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

Parameters

s1 First string to compare.

s2 Second string to compare.

Returns

True if collate<Char> facet compares *s1* < *s2*, else false.

5.533.4.7 `const locale& std::locale::operator= (const locale & __other) throw
()`

Assignment operator.

Set this locale to be a copy of *other*.

Parameters

other The locale to copy.

Returns

A reference to this locale.

5.533.4.8 `bool std::locale::operator==(const locale & __other) const throw ()`

Locale equality.

Parameters

other The locale to compare against.

Returns

True if *other* and this refer to the same locale instance, are copies, or have the same name. False otherwise.

5.533.5 Friends And Related Function Documentation

5.533.5.1 `template<typename _Facet > bool has_facet (const locale &)
throw () [friend]`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if locale contains a facet of type `Facet`, else false.

5.533.5.2 `template<typename _Facet > const _Facet& use_facet (const locale
&) [friend]`

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

Parameters

Facet The facet type to access.

locale The locale to use.

Returns

Reference to facet of type `Facet`.

Exceptions

`std::bad_cast` if locale doesn't contain a facet of type `Facet`.

5.533.6 Member Data Documentation

5.533.6.1 `const category std::locale::all` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

5.533.6.2 `const category std::locale::collate` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

5.533.6.3 `const category std::locale::ctype` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

5.533.6.4 `const category std::locale::messages` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

5.533.6.5 const category std::locale::monetary [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

5.533.6.6 const category std::locale::none [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

5.533.6.7 const category std::locale::numeric [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

5.533.6.8 const category std::locale::time [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

- [locale_classes.h](#)

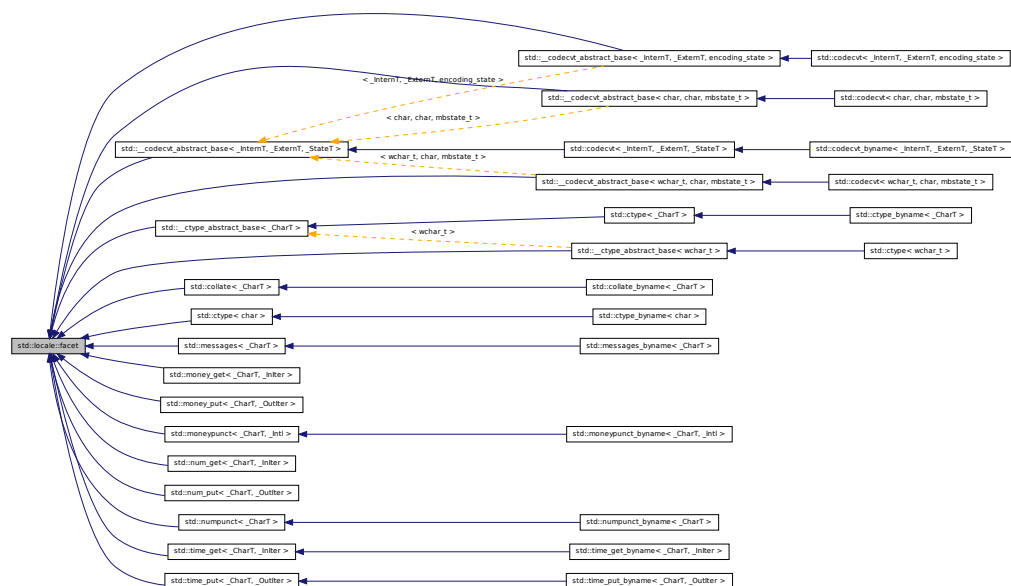
- [locale_classes.tcc](#)

5.534 std::locale::facet Class Reference

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Inheritance diagram for std::locale::facet:



Protected Member Functions

- `facet` (`size_t __refs=0`) `throw ()`
- `virtual ~facet ()`

Static Protected Member Functions

- `static __c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- `static void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`

- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Friends

- class **locale**
- class **locale::_Impl**

5.534.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management. Facets may not be copied or assigned.

Definition at line 337 of file locale_classes.h.

5.534.2 Constructor & Destructor Documentation

5.534.2.1 **std::locale::facet::facet (size_t __refs = 0) throw () [inline, explicit, protected]**

Facet constructor.

This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

Parameters

refs The initial value for reference count.

Definition at line 369 of file locale_classes.h.

5.534.2.2 **virtual std::locale::facet::~~facet () [protected, virtual]**

Facet destructor.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

5.535 std::locale::id Class Reference

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Public Member Functions

- `id ()`
- `size_t _M_id () const throw ()`

Friends

- `template<typename _Facet >`
`bool has_facet (const locale &) throw ()`
- `class locale`
- `class locale::_Impl`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &)`

5.535.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Definition at line 432 of file `locale_classes.h`.

5.535.2 Constructor & Destructor Documentation

5.535.2.1 std::locale::id::id () [inline]

Constructor.

Definition at line 463 of file `locale_classes.h`.

5.535.3 Friends And Related Function Documentation

5.535.3.1 `template<typename _Facet > bool has_facet (const locale &)
throw () [friend]`

Test for the presence of a facet.

has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if locale contains a facet of type Facet, else false.

5.535.3.2 `template<typename _Facet > const _Facet& use_facet (const locale
&) [friend]`

Return a facet.

use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws [std::bad_cast](#) if the locale doesn't contain a facet of type Facet.

Parameters

Facet The facet type to access.

locale The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

[std::bad_cast](#) if locale doesn't contain a facet of type Facet.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

5.536 `std::lock_guard< _Mutex >` Class Template Reference

Scoped lock idiom.

Public Types

- `typedef _Mutex mutex_type`

Public Member Functions

- `lock_guard` (`mutex_type &__m`)
- `lock_guard` (`mutex_type &__m`, [adopt_lock_t](#))
- `lock_guard` (`const lock_guard &`)
- `lock_guard` & `operator=` (`const lock_guard &`)

5.536.1 Detailed Description

`template<typename _Mutex> class std::lock_guard< _Mutex >`

Scoped lock idiom.

Definition at line 392 of file `mutex`.

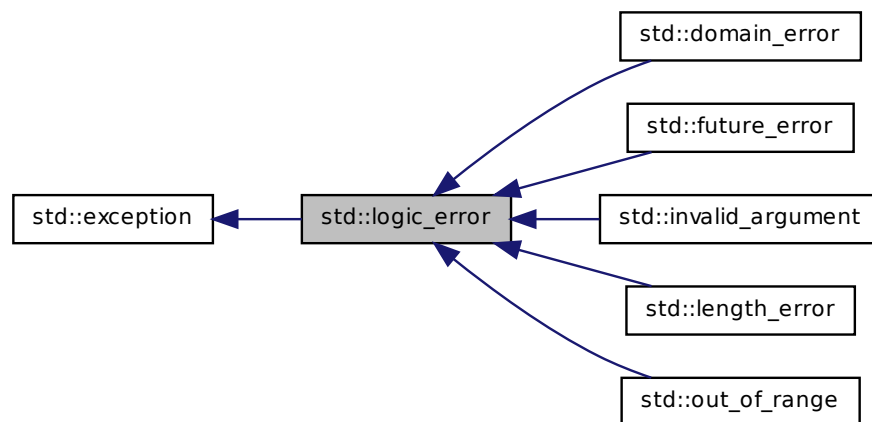
The documentation for this class was generated from the following file:

- [mutex](#)

5.537 `std::logic_error` Class Reference

One of two subclasses of exception.

Inheritance diagram for std::logic_error:



Public Member Functions

- `logic_error` (const `string` & __arg)
- virtual const char * `what` () const throw ()

5.537.1 Detailed Description

One of two subclasses of exception. Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 53 of file `stdexcept`.

5.537.2 Constructor & Destructor Documentation

5.537.2.1 `std::logic_error::logic_error (const string & __arg) [explicit]`

Takes a character string describing the error.

5.537.3 Member Function Documentation

5.537.3.1 `virtual const char* std::logic_error::what () const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

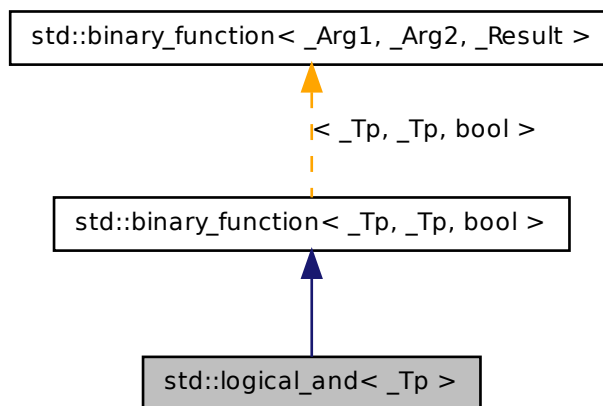
The documentation for this class was generated from the following file:

- [stdexcept](#)

5.538 `std::logical_and< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for `std::logical_and< _Tp >`:



Public Types

- typedef _Tp [first_argument_type](#)
- typedef bool [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.538.1 Detailed Description

template<typename _Tp> struct std::logical_and< _Tp >

One of the [Boolean operations functors](#).

Definition at line 263 of file stl_function.h.

5.538.2 Member Typedef Documentation

5.538.2.1 **typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.538.2.2 **typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]**

type of the return type

Definition at line 118 of file stl_function.h.

5.538.2.3 **typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]**

the type of the second argument

Definition at line 117 of file stl_function.h.

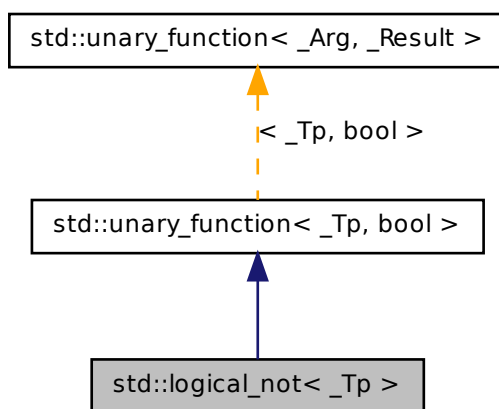
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.539 `std::logical_not< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for `std::logical_not< _Tp >`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `bool` [result_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x) const`

5.539.1 Detailed Description

`template<typename _Tp> struct std::logical_not< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 281 of file `stl_function.h`.

5.539.2 Member Typedef Documentation

5.539.2.1 `typedef _Tp std::unary_function< _Tp , bool >::argument_type` [`inherited`]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.539.2.2 `typedef bool std::unary_function< _Tp , bool >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

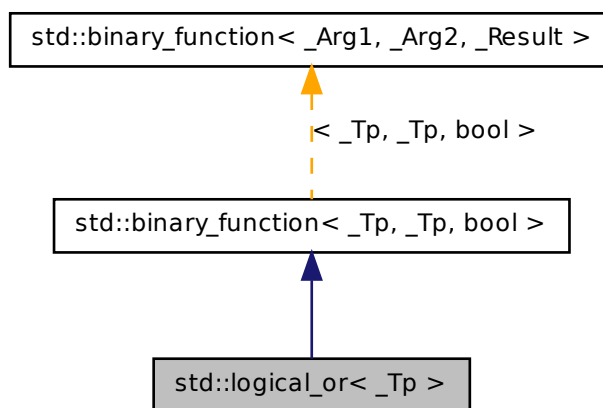
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.540 `std::logical_or< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for `std::logical_or< _Tp >`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.540.1 Detailed Description

`template<typename _Tp> struct std::logical_or< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 272 of file `stl_function.h`.

5.541 std::lognormal_distribution< _RealType > Class Template Reference 2651

5.540.2 Member Typedef Documentation

5.540.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.540.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.540.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.541 std::lognormal_distribution< _RealType > Class Template Reference

A [lognormal_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **lognormal_distribution** (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- **lognormal_distribution** (`const param_type &__p`)
- `_RealType m` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `void param` (`const param_type &__param`)
- `param_type param` () const
- `void reset` ()
- `_RealType s` () const

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & operator<<` (`std::basic_ostream< _CharT, _Traits > &, const std::lognormal_distribution< _RealType1 > &`)
- `template<typename _RealType1 > bool operator==` (`const std::lognormal_distribution< _RealType1 > &__d1, const std::lognormal_distribution< _RealType1 > &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream< _CharT, _Traits > &, std::lognormal_distribution< _RealType1 > &`)

5.541.1 Detailed Description

`template<typename _RealType = double> class std::lognormal_distribution< _RealType >`

A [lognormal_distribution](#) random number distribution. The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2174 of file random.h.

5.541.2 Member Typedef Documentation

5.541.2.1 `template<typename _RealType = double> typedef _RealType
std::lognormal_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2181 of file random.h.

5.541.3 Member Function Documentation

5.541.3.1 `template<typename _RealType = double> result_type
std::lognormal_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2265 of file random.h.

5.541.3.2 `template<typename _RealType = double> result_type
std::lognormal_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2258 of file random.h.

5.541.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::lognormal_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2273 of file random.h.

References `std::lognormal_distribution< _RealType >::operator()()`, and
`std::lognormal_distribution< _RealType >::param()`.

Referenced by `std::lognormal_distribution< _RealType >::operator()()`.

5.541.3.4 `template<typename _RealType = double> void
std::lognormal_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 2251 of file random.h.

5.541.3.5 `template<typename _RealType = double> param_type
std::lognormal_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2243 of file random.h.

Referenced by `std::lognormal_distribution< _RealType >::operator()()`.

5.541.3.6 `template<typename _RealType = double> void
std::lognormal_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2225 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

5.541.4 Friends And Related Function Documentation

5.541.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::lognormal_distribution< _RealType1 > &) [friend]`

Inserts a `lognormal_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

__os An output stream.

__x A `lognormal_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.542 `std::lognormal_distribution<_RealType>::param_type` Struct Reference 2655

5.541.4.2 `template<typename _RealType = double> template<typename _RealType1 > bool operator==(const std::lognormal_distribution<_RealType1 > & __d1, const std::lognormal_distribution<_RealType1 > & __d2) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2289 of file random.h.

5.541.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>>(std::basic_istream<_CharT, _Traits> & , std::lognormal_distribution<_RealType1> &) [friend]`

Extracts a `lognormal_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `lognormal_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.542 `std::lognormal_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [lognormal_distribution](#)<_RealType> **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- _RealType **m** () const
- _RealType **s** () const

Friends

- `bool operator==(const param_type &__p1, const param_type &__p2)`

5.542.1 Detailed Description

`template<typename _RealType = double> struct std::lognormal_distribution<
_RealType >::param_type`

Parameter type.

Definition at line 2183 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.543 `std::make_signed< _Tp >` Struct Template Reference

[make_signed](#)

Public Types

- `typedef __make_signed_selector< _Tp >::__type type`

5.543.1 Detailed Description

`template<typename _Tp> struct std::make_signed< _Tp >`

[make_signed](#)

Definition at line 644 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.544 `std::make_unsigned< _Tp >` Struct Template Reference

[make_unsigned](#)

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2657

Public Types

- typedef __make_unsigned_selector< _Tp >::__type **type**

5.544.1 Detailed Description

template<typename _Tp> struct std::make_unsigned< _Tp >

[make_unsigned](#)

Definition at line 567 of file type_traits.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Pair_alloc_type::const_pointer **const_pointer**
- typedef _Pair_alloc_type::const_reference **const_reference**
- typedef [_Rep_type::const_reverse_iterator](#) **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Pair_alloc_type::pointer **pointer**
- typedef _Pair_alloc_type::reference **reference**
- typedef [_Rep_type::reverse_iterator](#) **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef [std::pair](#)< const _Key, _Tp > **value_type**

Public Member Functions

- [map](#) ()
- [map](#) (const [_Compare](#) &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- [map](#) ([map](#) &&__x)
- [map](#) ([initializer_list](#)< [value_type](#) > __l, const [_Compare](#) &__c=[_Compare](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- [map](#) (const [map](#) &__x)
- [template](#)<typename [_InputIterator](#) >
[map](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)
- [template](#)<typename [_InputIterator](#) >
[map](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, const [_Compare](#) &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- [mapped_type](#) & [at](#) (const [key_type](#) &__k)
- const [mapped_type](#) & [at](#) (const [key_type](#) &__k) const
- [iterator](#) [begin](#) ()
- const [iterator](#) [begin](#) () const
- const [iterator](#) [cbegin](#) () const
- const [iterator](#) [cend](#) () const
- void [clear](#) ()
- [size_type](#) [count](#) (const [key_type](#) &__x) const
- [const_reverse_iterator](#) [crbegin](#) () const
- [const_reverse_iterator](#) [crend](#) () const
- bool [empty](#) () const
- [iterator](#) [end](#) ()
- const [iterator](#) [end](#) () const
- [std::pair](#)< const [iterator](#), const [iterator](#) > [equal_range](#) (const [key_type](#) &__x) const
- [std::pair](#)< [iterator](#), [iterator](#) > [equal_range](#) (const [key_type](#) &__x)
- [iterator](#) [erase](#) ([iterator](#) __position)
- [size_type](#) [erase](#) (const [key_type](#) &__x)
- [iterator](#) [erase](#) ([iterator](#) __first, [iterator](#) __last)
- [iterator](#) [find](#) (const [key_type](#) &__x)
- const [iterator](#) [find](#) (const [key_type](#) &__x) const
- [allocator_type](#) [get_allocator](#) () const
- [template](#)<typename [_InputIterator](#) >
void [insert](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)
- [iterator](#) [insert](#) ([iterator](#) __position, const [value_type](#) &__x)
- [std::pair](#)< [iterator](#), bool > [insert](#) (const [value_type](#) &__x)
- void [insert](#) ([std::initializer_list](#)< [value_type](#) > __list)
- [key_compare](#) [key_comp](#) () const
- const [iterator](#) [lower_bound](#) (const [key_type](#) &__x) const
- [iterator](#) [lower_bound](#) (const [key_type](#) &__x)

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2659

- size_type [max_size](#) () const
- [map](#) & [operator=](#) ([map](#) &&__x)
- [map](#) & [operator=](#) (const [map](#) &__x)
- [map](#) & [operator=](#) (initializer_list< value_type > __l)
- mapped_type & [operator\[\]](#) (const key_type &__k)
- [const_reverse_iterator](#) [rbegin](#) () const
- [reverse_iterator](#) [rbegin](#) ()
- [reverse_iterator](#) [rend](#) ()
- [const_reverse_iterator](#) [rend](#) () const
- size_type [size](#) () const
- void [swap](#) ([map](#) &__x)
- iterator [upper_bound](#) (const key_type &__x)
- const_iterator [upper_bound](#) (const key_type &__x) const
- value_compare [value_comp](#) () const

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool [operator](#)< (const [map](#)< _K1, _T1, _C1, _A1 > &, const [map](#)< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool [operator==](#) (const [map](#)< _K1, _T1, _C1, _A1 > &, const [map](#)< _K1, _T1, _C1, _A1 > &)

5.545.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class std::map< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 86 of file `stl_map.h`.

5.545.2 Constructor & Destructor Documentation

5.545.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map () [inline]`

Default constructor creates no elements.

Definition at line 150 of file `stl_map.h`.

5.545.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map (const _Compare & __comp, const allocator_type & __a = allocator_type()) [inline, explicit]`

Creates a map with no elements.

Parameters

comp A comparison object.

a An allocator object.

Definition at line 159 of file `stl_map.h`.

5.545.2.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map (const map<_Key, _Tp, _Compare, _Alloc> & __x) [inline]`

Map copy constructor.

Parameters

x A map of identical element and allocator types.

The newly-created map uses a copy of the allocation object used by *x*.

Definition at line 170 of file `stl_map.h`.

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2661

5.545.2.4 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (
map< _Key, _Tp, _Compare, _Alloc > && __x) [inline]`

Map move constructor.

Parameters

x A map of identical element and allocator types.

The newly-created map contains the exact contents of *x*. The contents of *x* are a valid, but unspecified map.

Definition at line 181 of file stl_map.h.

5.545.2.5 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (
initializer_list< value_type > __l, const _Compare & __c =
_Compare(), const allocator_type & __a = allocator_type()
) [inline]`

Builds a map from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison object.

a An allocator object.

Create a map consisting of copies of the elements in the [initializer_list](#) *l*. This is linear in N if the range is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 195 of file stl_map.h.

5.545.2.6 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> template<typename _InputIterator > std::map<
_Key, _Tp, _Compare, _Alloc >::map (_InputIterator __first,
_InputIterator __last) [inline]`

Builds a map from a range.

Parameters

first An input iterator.

last An input iterator.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 212 of file stl_map.h.

```
5.545.2.7  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> template<typename _InputIterator > std::map<
_Key, _Tp, _Compare, _Alloc >::map ( _InputIterator __first,
_InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a map from a range.

Parameters

first An input iterator.
last An input iterator.
comp A comparison functor.
a An allocator object.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 228 of file stl_map.h.

5.545.3 Member Function Documentation

```
5.545.3.1  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> mapped_type& std::map<_Key, _Tp, _Compare,
_Alloc >::at ( const key_type & __k ) [inline]
```

Access to map data.

Parameters

k The key for which data should be retrieved.

Returns

A reference to the data whose key is equivalent to *k*, if such a data is present in the map.

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2663

Exceptions

std::out_of_range If no such data is present.

Definition at line 465 of file stl_map.h.

5.545.3.2 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc
>::begin () [inline]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 307 of file stl_map.h.

5.545.3.3 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 316 of file stl_map.h.

5.545.3.4 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 380 of file stl_map.h.

5.545.3.5 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 389 of file stl_map.h.

5.545.3.6 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc
>::clear () [inline]`

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 663 of file `stl_map.h`.

5.545.3.7 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc
>::count (const key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

`x` Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 723 of file `stl_map.h`.

5.545.3.8 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp,
_Compare, _Alloc >::crbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 398 of file `stl_map.h`.

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2665

5.545.3.9 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp,
_Compare, _Alloc >::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 407 of file stl_map.h.

5.545.3.10 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> bool std::map< _Key, _Tp, _Compare, _Alloc
>::empty () const [inline]`

Returns true if the map is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Definition at line 416 of file stl_map.h.

5.545.3.11 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 334 of file stl_map.h.

5.545.3.12 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc
>::end () [inline]`

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 325 of file stl_map.h.

5.545.3.13 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> std::pair<iterator, iterator> std::map<_Key,
_Tp, _Compare, _Alloc >::equal_range (const key_type & __x)
[inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 792 of file stl_map.h.

5.545.3.14 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> std::pair<const_iterator, const_iterator>
std::map<_Key, _Tp, _Compare, _Alloc >::equal_range (const
key_type & __x) const [inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2667

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 811 of file stl_map.h.

```
5.545.3.15  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
             >::erase ( iterator __position ) [inline]
```

Erases an element from a map.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, [end\(\)](#) is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 573 of file stl_map.h.

```
5.545.3.16  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc  
             >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 603 of file stl_map.h.

5.545.3.17 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::erase (iterator __first, iterator __last) [inline]`

Erases a [first,last) range of elements from a map.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 622 of file stl_map.h.

5.545.3.18 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::find (const key_type & __x) const [inline]`

Tries to locate an element in a map.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 711 of file stl_map.h.

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2669

5.545.3.19 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc
>::find (const key_type & __x) [inline]`

Tries to locate an element in a map.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 696 of file `stl_map.h`.

5.545.3.20 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> allocator_type std::map< _Key, _Tp, _Compare,
_Alloc >::get_allocator () const [inline]`

Get a copy of the memory allocation object.

Definition at line 297 of file `stl_map.h`.

5.545.3.21 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::pair<iterator, bool> std::map< _Key,
_Tp, _Compare, _Alloc >::insert (const value_type & __x)
[inline]`

Attempts to insert a `std::pair` into the map.

Parameters

x Pair to be inserted (see `std::make_pair` for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 500 of file `stl_map.h`.

```
5.545.3.22  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc
              >::insert ( std::initializer_list< value_type > __list ) [inline]
```

Attempts to insert a list of `std::pairs` into the map.

Parameters

list A `std::initializer_list<value_type>` of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 512 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::insert()`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
5.545.3.23  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc
              >::insert ( iterator __position, const value_type & __x )
              [inline]
```

Attempts to insert a [std::pair](#) into the map.

Parameters

position An iterator that serves as a hint as to where the pair should be inserted.

x Pair to be inserted (see [std::make_pair](#) for easy creation of pairs).

Returns

An iterator that points to the element with key of *x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument [insert\(\)](#) does. Note that the first parameter

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2671

is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 540 of file stl_map.h.

```
5.545.3.24  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> template<typename _InputIterator > void
              std::map< _Key, _Tp, _Compare, _Alloc >::insert ( _InputIterator
              __first, _InputIterator __last ) [inline]
```

Template function that attempts to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 553 of file stl_map.h.

```
5.545.3.25  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> key_compare std::map< _Key, _Tp, _Compare,
              _Alloc >::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 672 of file stl_map.h.

```
5.545.3.26  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc
              >::lower_bound ( const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 738 of file `stl_map.h`.

```
5.545.3.27  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare,
              _Alloc >::lower_bound ( const key_type & __x ) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 753 of file `stl_map.h`.

```
5.545.3.28  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc
              >::max_size ( ) const  [inline]
```

Returns the maximum size of the map.

Definition at line 426 of file `stl_map.h`.

```
5.545.3.29  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> map& std::map< _Key, _Tp, _Compare, _Alloc
              >::operator= ( map< _Key, _Tp, _Compare, _Alloc > && __x )
              [inline]
```

Map move assignment operator.

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2673

Parameters

x A map of identical element and allocator types.

The contents of *x* are moved into this map (without copying). *x* is a valid, but unspecified map.

Definition at line 266 of file stl_map.h.

```
5.545.3.30 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc  
>::operator= ( const map< _Key, _Tp, _Compare, _Alloc > & __x  
) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

x A map of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 251 of file stl_map.h.

```
5.545.3.31 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc  
>::operator= ( initializer_list< value_type > __l ) [inline]
```

Map list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a map with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 287 of file stl_map.h.

5.545.3.32 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map<_Key, _Tp, _Compare, _Alloc>::operator[] (const key_type & __k) [inline]`

Subscript (`[]`) access to map data.

Parameters

k The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript (`[]`) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 443 of file `stl_map.h`.

5.545.3.33 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 343 of file `stl_map.h`.

5.545.3.34 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 352 of file `stl_map.h`.

5.545 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2675

5.545.3.35 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp,
_Compare, _Alloc >::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 370 of file stl_map.h.

5.545.3.36 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 361 of file stl_map.h.

5.545.3.37 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc
>::size () const [inline]`

Returns the size of the map.

Definition at line 421 of file stl_map.h.

5.545.3.38 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc
>::swap (map< _Key, _Tp, _Compare, _Alloc > & __x)
[inline]`

Swaps data with another map.

Parameters

x A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 653 of file stl_map.h.

Referenced by std::swap().

```
5.545.3.39  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare,
              _Alloc>::upper_bound ( const key_type & __x ) const
              [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 773 of file stl_map.h.

```
5.545.3.40  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc
              >::upper_bound ( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 763 of file stl_map.h.

```
5.545.3.41  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> value_compare std::map<_Key, _Tp, _Compare,
              _Alloc>::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 680 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl_map.h](#)

5.546 `std::mask_array< _Tp >` Class Template Reference

Reference to selected subset of an array.

Public Types

- `typedef _Tp value_type`

Public Member Functions

- [mask_array](#) (const [mask_array](#) &)
- `template<class _Dom >`
void **operator%=** (const `_Expr< _Dom, _Tp >` &) const
- void [operator%=](#) (const [valarray](#)< _Tp > &) const
- void [operator&=](#) (const [valarray](#)< _Tp > &) const
- `template<class _Dom >`
void **operator&=** (const `_Expr< _Dom, _Tp >` &) const
- `template<class _Dom >`
void **operator*=** (const `_Expr< _Dom, _Tp >` &) const
- void [operator*=](#) (const [valarray](#)< _Tp > &) const
- `template<class _Dom >`
void **operator+=** (const `_Expr< _Dom, _Tp >` &) const
- void [operator+=](#) (const [valarray](#)< _Tp > &) const
- void [operator-=](#) (const [valarray](#)< _Tp > &) const
- `template<class _Dom >`
void **operator-=** (const `_Expr< _Dom, _Tp >` &) const
- `template<class _Dom >`
void **operator/=** (const `_Expr< _Dom, _Tp >` &) const
- void [operator/=](#) (const [valarray](#)< _Tp > &) const
- void [operator<<=](#) (const [valarray](#)< _Tp > &) const
- `template<class _Dom >`
void **operator<<=** (const `_Expr< _Dom, _Tp >` &) const
- `template<class _Dom >`
void **operator=** (const `_Expr< _Dom, _Tp >` &) const

- `mask_array & operator= (const mask_array &)`
- `void operator= (const valarray<_Tp> &) const`
- `template<class _Ex>`
`void operator= (const _Expr<_Ex, _Tp> &__e) const`
- `void operator= (const _Tp &) const`
- `void operator>>= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator>>= (const _Expr<_Dom, _Tp> &) const`
- `void operator^= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator^= (const _Expr<_Dom, _Tp> &) const`
- `void operator|= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator|= (const _Expr<_Dom, _Tp> &) const`

Friends

- `class valarray<_Tp>`

5.546.1 Detailed Description

`template<class _Tp> class std::mask_array<_Tp>`

Reference to selected subset of an array. A `mask_array` is a reference to the actual elements of an array specified by a bitmask in the form of an array of `bool`. The way to get a `mask_array` is to call `operator[](valarray<bool>)` on a `valarray`. The returned `mask_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if a `mask_array` is obtained using the array `(false, true, false, true)` as an argument, the mask array has two elements referring to `array[1]` and `array[3]` in the underlying array.

Parameters

Tp Element type.

Definition at line 61 of file `mask_array.h`.

5.546.2 Member Function Documentation

5.546.2.1 `template<class _Tp> void std::mask_array<_Tp>::operator%=(const valarray<_Tp> &) const`

Modulo slice elements by corresponding elements of *v*.

5.546.2.2 `template<class _Tp> void std::mask_array< _Tp >::operator&= (const valarray< _Tp > &) const`

Logical and slice elements with corresponding elements of *v*.

5.546.2.3 `template<class _Tp> void std::mask_array< _Tp >::operator*= (const valarray< _Tp > &) const`

Multiply slice elements by corresponding elements of *v*.

5.546.2.4 `template<class _Tp> void std::mask_array< _Tp >::operator+= (const valarray< _Tp > &) const`

Add corresponding elements of *v* to slice elements.

5.546.2.5 `template<class _Tp> void std::mask_array< _Tp >::operator-= (const valarray< _Tp > &) const`

Subtract corresponding elements of *v* from slice elements.

5.546.2.6 `template<class _Tp> void std::mask_array< _Tp >::operator/= (const valarray< _Tp > &) const`

Divide slice elements by corresponding elements of *v*.

5.546.2.7 `template<class _Tp> void std::mask_array< _Tp >::operator<<= (const valarray< _Tp > &) const`

Left shift slice elements by corresponding elements of *v*.

5.546.2.8 `template<class _Tp> void std::mask_array< _Tp >::operator>>= (const valarray< _Tp > &) const`

Right shift slice elements by corresponding elements of *v*.

5.546.2.9 `template<class _Tp> void std::mask_array< _Tp >::operator^= (const valarray< _Tp > &) const`

Logical xor slice elements with corresponding elements of *v*.

5.546.2.10 `template<class _Tp> void std::mask_array< _Tp >::operator|= (const valarray< _Tp > &) const`

Logical or slice elements with corresponding elements of v.

The documentation for this class was generated from the following file:

- [mask_array.h](#)

5.547 std::match_results< _Bi_iter, _Allocator > Class Template Reference

The results of a match or search operation.

Inheritance diagram for std::match_results< _Bi_iter, _Allocator >:



Private Types

- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**

Private Member Functions

- `_Tp_alloc_type::pointer` **_M_allocate** (size_t __n)
- `pointer` **_M_allocate_and_copy** (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, `std::input_iterator_tag`)
- void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, `std::forward_iterator_tag`)
- void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, `_false_type`)
- void **_M_assign_dispatch** (_Integer __n, _Integer __val, `_true_type`)
- `size_type` **_M_check_len** (size_type __n, const char *__s) const
- void **_M_deallocate** (typename `_Tp_alloc_type::pointer` __p, size_t __n)
- void **_M_default_append** (size_type __n)

- `void _M_default_initialize (size_type __n)`
- `void _M_erase_at_end (pointer __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `void _M_initialize_dispatch (_Integer __n, _Integer __value, __true_type)`
- `void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_insert_aux (iterator __position, _Args &&... __args)`
- `void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `void _M_range_insert (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (size_type __n, const value_type &__val)`
- `void assign (initializer_list< value_type > __l)`
- `reference at (size_type __n)`
- `const_reference at (size_type __n) const`
- `reference back ()`
- `const_reference back () const`
- `iterator begin ()`
- `size_type capacity () const`
- `void clear ()`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `std::sub_match<_Bi_iter> * data ()`
- `const std::sub_match<_Bi_iter> * data () const`
- `iterator emplace (iterator __position, _Args &&... __args)`
- `void emplace_back (_Args &&... __args)`
- `iterator end ()`
- `iterator erase (iterator __position)`
- `iterator erase (iterator __first, iterator __last)`

- [reference front](#) ()
- [const_reference front](#) () const
- void [insert](#) (iterator __position, _InputIterator __first, _InputIterator __last)
- iterator [insert](#) (iterator __position, [value_type](#) &&__x)
- void [insert](#) (iterator __position, size_type __n, const [value_type](#) &__x)
- iterator [insert](#) (iterator __position, const [value_type](#) &__x)
- void [insert](#) (iterator __position, [initializer_list](#)< [value_type](#) > __l)
- [reference operator\[\]](#) (size_type __n)
- [const_reference operator\[\]](#) (size_type __n) const
- void [pop_back](#) ()
- void [push_back](#) (const [value_type](#) &__x)
- void [push_back](#) ([value_type](#) &&__x)
- [const_reverse_iterator rbegin](#) () const
- [reverse_iterator rbegin](#) ()
- [reverse_iterator rend](#) ()
- [const_reverse_iterator rend](#) () const
- void [reserve](#) (size_type __n)
- void [resize](#) (size_type __new_size, const [value_type](#) &__x)
- void [resize](#) (size_type __new_size)
- void [shrink_to_fit](#) ()
- void [swap](#) ([vector](#) &__x)

Private Attributes

- [_Vector_impl _M_impl](#)

Friends

- class [__regex::_SpecializedResults](#)< [_Bi_iter](#), [_Allocator](#) >

10.? Public Types

- typedef [sub_match](#)< [_Bi_iter](#) > [value_type](#)
- typedef const [value_type](#) & [const_reference](#)
- typedef [const_reference](#) [reference](#)
- typedef [_Base_type](#)::const_iterator [const_iterator](#)
- typedef const_iterator [iterator](#)
- typedef [std::iterator_traits](#)< [_Bi_iter](#) >::difference_type [difference_type](#)
- typedef [_Allocator](#)::size_type [size_type](#)
- typedef [_Allocator](#) [allocator_type](#)
- typedef [std::iterator_traits](#)< [_Bi_iter](#) >::[value_type](#) [char_type](#)
- typedef [std::basic_string](#)< [char_type](#) > [string_type](#)

10.1 Construction, Copying, and Destruction

- `match_results` (`const _Allocator &__a=_Allocator()`)
- `match_results` (`const match_results &__rhs`)
- `match_results &operator=` (`const match_results __rhs`)
- `~match_results` ()

10.2 Size

- `size_type size` () `const`
- `size_type max_size` () `const`
- `bool empty` () `const`

10.3 Element Access

- `difference_type length` (`size_type __sub=0`) `const`
- `difference_type position` (`size_type __sub=0`) `const`
- `string_type str` (`size_type __sub=0`) `const`
- `const_reference operator[]` (`size_type __sub`) `const`
- `const_reference prefix` () `const`
- `const_reference suffix` () `const`
- `const_iterator begin` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator end` () `const`
- `const_iterator cend` () `const`

10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter>`
`_Out_iter format` (`_Out_iter __out, const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::format_default`) `const`
- `string_type format` (`const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::format_default`) `const`

10.5 Allocator

- `allocator_type get_allocator` () `const`

10.6 Swap

- void [swap](#) ([match_results](#) &__that)

5.547.1 Detailed Description

template<typename [_Bi_iter](#), typename [_Allocator](#) = [allocator](#)<[sub_match](#)<[_Bi_iter](#)> >> class [std::match_results](#)< [_Bi_iter](#), [_Allocator](#) >

The results of a match or search operation. A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template [match_results](#).

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The [sub_match](#) object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the [sub_match](#) member [matched](#) is always true. The [sub_match](#) object stored at index *n* denotes what matched the marked sub-expression *n* within the matched expression. If the sub-expression *n* participated in a regular expression match then the [sub_match](#) member [matched](#) evaluates to true, and members [first](#) and [second](#) denote the range of characters [[first](#), [second](#)) which formed that match. Otherwise [matched](#) is false, and members [first](#) and [second](#) point to the end of the sequence that was searched.

Definition at line 1446 of file [regex.h](#).

5.547.2 Constructor & Destructor Documentation

5.547.2.1 `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter> >> std::match_results<_Bi_iter, _Allocator >::match_results (const _Allocator & __a = _Allocator()) [inline, explicit]`

Constructs a default [match_results](#) container.

Postcondition

[size\(\)](#) returns 0 and [str\(\)](#) returns an empty string.

Definition at line 1495 of file [regex.h](#).

Referenced by [std::match_results< \[_FwdIterT\]\(#\), \[_Alloc\]\(#\) >::operator=\(\)](#).

5.547 std::match_results< _Bi_iter, _Allocator > Class Template Reference 2685

5.547.2.2 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter,
_Allocator >::match_results (const match_results< _Bi_iter,
_Allocator > & __rhs) [inline]`

Copy constructs a match_results.

Definition at line 1502 of file regex.h.

5.547.2.3 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter,
_Allocator >::~~match_results () [inline]`

Destroys a match_results object.

Definition at line 1519 of file regex.h.

5.547.3 Member Function Documentation

5.547.3.1 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_iterator
std::match_results< _Bi_iter, _Allocator >::begin () const
[inline]`

Gets an iterator to the start of the sub_match collection.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1658 of file regex.h.

5.547.3.2 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_iterator
std::match_results< _Bi_iter, _Allocator >::cbegin () const
[inline]`

Gets an iterator to the start of the sub_match collection.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1665 of file regex.h.

5.547.3.3 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_iterator
std::match_results< _Bi_iter, _Allocator >::cend () const
[inline]`

Gets an iterator to one-past-the-end of the collection.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1683 of file regex.h.

5.547.3.4 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> bool std::match_results<
_Bi_iter, _Allocator >::empty () const [inline]`

Indicates if the match_results contains no results.

Return values

true The match_results object is empty.

false The match_results object is not empty.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1555 of file regex.h.

Referenced by `std::match_results< _FwdIterT, _Alloc >::cend()`, `std::match_results< _FwdIterT, _Alloc >::end()`, `std::match_results< _FwdIterT, _Alloc >::prefix()`, and `std::match_results< _FwdIterT, _Alloc >::suffix()`.

5.547.3.5 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_iterator
std::match_results< _Bi_iter, _Allocator >::end () const
[inline]`

Gets an iterator to one-past-the-end of the collection.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1672 of file regex.h.

```
5.547.3.6 template<typename _Bi_iter, typename _Allocator
= allocator<sub_match<_Bi_iter> >> string_type
std::match_results< _Bi_iter, _Allocator >::format ( const
string_type & __fmt, regex_constants::match_flag_type __flags =
regex_constants::format_default ) const
```

Todo

Implement this function.

```
5.547.3.7 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> template<typename
_Out_iter > _Out_iter std::match_results< _Bi_iter,
_Allocator >::format ( _Out_iter __out, const string_type
& __fmt, regex_constants::match_flag_type __flags =
regex_constants::format_default ) const [inline]
```

Todo

Implement this function.

Definition at line 1707 of file regex.h.

```
5.547.3.8 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> allocator_type
std::match_results< _Bi_iter, _Allocator >::get_allocator ( ) const
[inline]
```

Gets a copy of the allocator.

Reimplemented from [std::_Vector_base< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1731 of file regex.h.

```
5.547.3.9 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> difference_type
std::match_results< _Bi_iter, _Allocator >::length ( size_type __sub
= 0 ) const [inline]
```

Gets the length of the indicated submatch.

Parameters

sub indicates the submatch.

This function returns the length of the indicated submatch, or the length of the entire match if `sub` is zero (the default).

Definition at line 1573 of file `regex.h`.

5.547.3.10 `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> size_type std::match_results<_Bi_iter, _Allocator>::max_size() const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Reimplemented from `std::vector< std::sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1546 of file `regex.h`.

5.547.3.11 `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> match_results& std::match_results<_Bi_iter, _Allocator>::operator=(const match_results<_Bi_iter, _Allocator> __rhs) [inline]`

Assigns `rhs` to `*this`.

Definition at line 1510 of file `regex.h`.

5.547.3.12 `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> const_reference std::match_results<_Bi_iter, _Allocator>::operator[](size_type __sub) const [inline]`

Gets a `sub_match` reference for the match or submatch.

Parameters

sub indicates the submatch.

This function gets a reference to the indicated submatch, or the entire match if `sub` is zero.

If `sub >= size()` then this function returns a `sub_match` with a special value indicating no submatch.

Definition at line 1617 of file `regex.h`.

5.547 std::match_results< _Bi_iter, _Allocator > Class Template Reference 2689

5.547.3.13 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> difference_type
std::match_results< _Bi_iter, _Allocator >::position (size_type
__sub = 0) const [inline]`

Gets the offset of the beginning of the indicated submatch.

Parameters

sub indicates the submatch.

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of *sub* is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if *sub* is out of range.

Definition at line 1589 of file regex.h.

5.547.3.14 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_reference
std::match_results< _Bi_iter, _Allocator >::prefix () const
[inline]`

Gets a sub_match representing the match prefix.

This function gets a reference to a sub_match object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1632 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::position().

5.547.3.15 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> size_type std::match_results<
_Bi_iter, _Allocator >::size () const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1539 of file regex.h.

Referenced by `std::match_results<_FwdIterT, _Alloc >::operator[]()`, and `std::match_results<_FwdIterT, _Alloc >::position()`.

```
5.547.3.16 template<typename _Bi_iter, typename _Allocator
              = allocator<sub_match<_Bi_iter> >> string_type
              std::match_results<_Bi_iter, _Allocator >::str ( size_type __sub =
              0 ) const [inline]
```

Gets the match or submatch converted to a string type.

Parameters

sub indicates the submatch.

This function gets the submatch (or match, if *sub* is zero) extracted from the target range and converted to the associated string type.

Definition at line 1603 of file regex.h.

```
5.547.3.17 template<typename _Bi_iter, typename _Allocator =
              allocator<sub_match<_Bi_iter> >> const_reference
              std::match_results<_Bi_iter, _Allocator >::suffix ( ) const
              [inline]
```

Gets a `sub_match` representing the match suffix.

This function gets a reference to a `sub_match` object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1647 of file regex.h.

```
5.547.3.18 template<typename _Bi_iter, typename _Allocator =
              allocator<sub_match<_Bi_iter> >> void std::match_results<
              _Bi_iter, _Allocator >::swap ( match_results<_Bi_iter, _Allocator
              > & __that ) [inline]
```

Swaps the contents of two [match_results](#).

Definition at line 1745 of file regex.h.

Referenced by `std::swap()`.

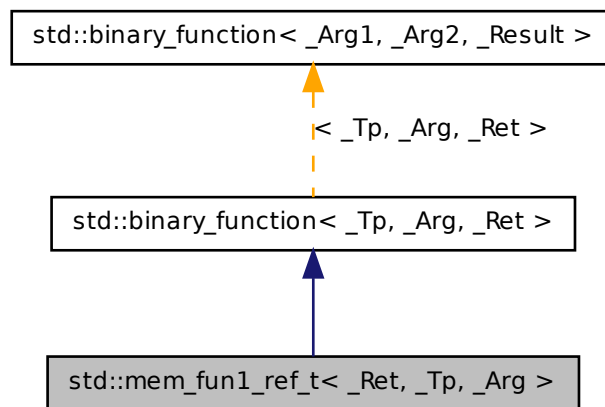
The documentation for this class was generated from the following file:

- [regex.h](#)

5.548 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `mem_fun1_ref_t` (`_Ret` (`_Tp::*__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp &__r, _Arg __x`) const

5.548.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg> class std::mem_
fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member /// pointers](#).

Definition at line 632 of file stl_function.h.

5.548.2 Member Typedef Documentation

5.548.2.1 `typedef _Tp std::binary_function< _Tp , _Arg , _Ret
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.548.2.2 `typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.548.2.3 `typedef _Arg std::binary_function< _Tp , _Arg , _Ret
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

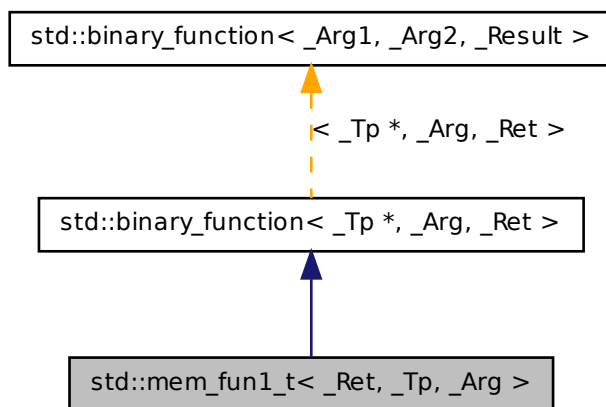
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.549 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef `_Tp *` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `mem_fun1_t` (`_Ret`(`_Tp::*__pf`)(`_Arg`))
- `_Ret operator()` (`_Tp *__p`, `_Arg __x`) const

5.549.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg> class std::mem_
fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member /// pointers](#).

Definition at line 596 of file `stl_function.h`.

5.549.2 Member Typedef Documentation

5.549.2.1 `typedef _Tp * std::binary_function< _Tp *, _Arg , _Ret
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.549.2.2 `typedef _Ret std::binary_function< _Tp *, _Arg , _Ret
>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.549.2.3 `typedef _Arg std::binary_function< _Tp *, _Arg , _Ret
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

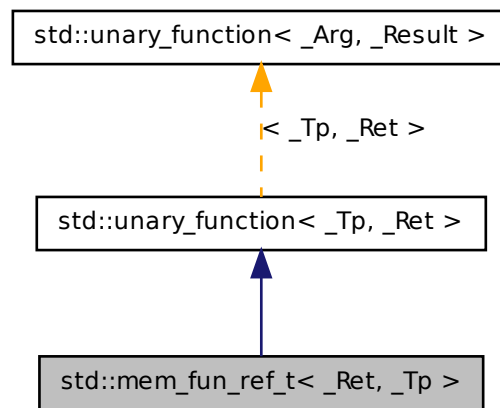
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.550 `std::mem_fun_ref_t< _Ret, _Tp >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun_ref_t<_Ret, _Tp>`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `mem_fun_ref_t` (`_Ret`(`_Tp::*__pf`)())
- `_Ret operator()` (`_Tp &__r`) const

5.550.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::mem_fun_ref_t<_Ret, _Tp>
```

>

One of the [adaptors for member /// pointers](#).

Definition at line 560 of file `stl_function.h`.

5.550.2 Member Typedef Documentation

5.550.2.1 `typedef _Tp std::unary_function< _Tp , _Ret >::argument_type` `[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.550.2.2 `typedef _Ret std::unary_function< _Tp , _Ret >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

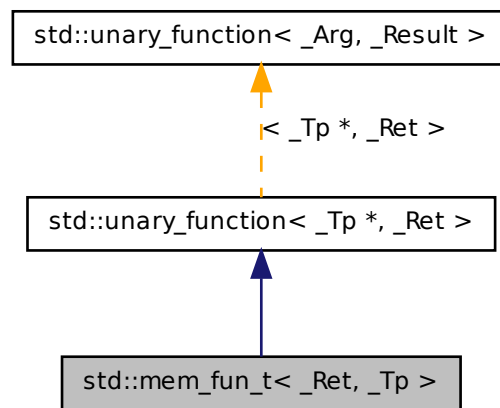
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.551 `std::mem_fun_t< _Ret, _Tp >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem_fun_t< _Ret, _Tp >:



Public Types

- typedef `_Tp *` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `mem_fun_t` (`_Ret`(`_Tp::*__pf`)())
- `_Ret operator()` (`_Tp *__p`) const

5.551.1 Detailed Description

template<typename `_Ret`, typename `_Tp`> class std::mem_fun_t< `_Ret`, `_Tp` >

One of the [adaptors for member /// pointers](#).

Definition at line 524 of file `stl_function.h`.

5.551.2 Member Typedef Documentation

5.551.2.1 `typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type` `[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.551.2.2 `typedef _Ret std::unary_function< _Tp *, _Ret >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

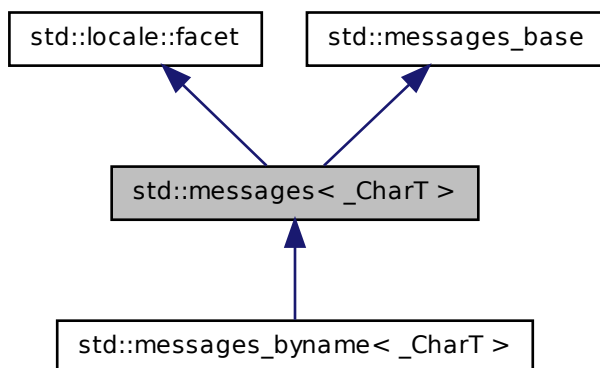
- [stl_function.h](#)

5.552 `std::messages< _CharT >` Class Template Reference

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

Inheritance diagram for std::messages<_CharT>:



Public Types

- typedef int **catalog**
- typedef `_CharT` **char_type**
- typedef `basic_string<_CharT>` **string_type**

Public Member Functions

- **messages** (size_t __refs=0)
- **messages** (__c_locale __cloc, const char *__s, size_t __refs=0)
- void **close** (catalog __c) const
- **string_type** **get** (catalog __c, int __set, int __msgid, const **string_type** &__s) const
- catalog **open** (const **basic_string**< char > &__s, const **locale** &__loc) const
- catalog **open** (const **basic_string**< char > &, const **locale** &, const char *) const

Static Public Attributes

- static **locale::id** **id**

Protected Member Functions

- virtual `~messages()`
- `string_type _M_convert_from_char` (char *) const
- char * `_M_convert_to_char` (const `string_type` &__msg) const
- virtual void `do_close` (catalog) const
- template<>
`string do_get` (catalog, int, int, const `string` &) const
- template<>
`wstring do_get` (catalog, int, int, const `wstring` &) const
- virtual `string_type do_get` (catalog, int, int, const `string_type` &__default) const
- virtual catalog `do_open` (const `basic_string`< char > &, const `locale` &) const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (__c_locale &__cloc) throw ()
- static void `_S_create_c_locale` (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void `_S_destroy_c_locale` (__c_locale &__cloc)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char * `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (__c_locale __cloc, const char *__s)

Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char * `_M_name_messages`

Friends

- class `locale::_Impl`

5.552.1 Detailed Description

`template<typename _CharT> class std::messages< _CharT >`

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined. This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The

second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1690 of file locale_facets_nonio.h.

5.552.2 Member Typedef Documentation

5.552.2.1 `template<typename _CharT> typedef _CharT std::messages<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::messages_byname<_CharT>](#).

Definition at line 1696 of file locale_facets_nonio.h.

5.552.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::messages<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::messages_byname<_CharT>](#).

Definition at line 1697 of file locale_facets_nonio.h.

5.552.3 Constructor & Destructor Documentation

5.552.3.1 `template<typename _CharT> std::messages<_CharT>::messages (size_t __refs = 0) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 43 of file messages_members.h.

5.552.3.2 `template<typename _CharT > std::messages< _CharT >::messages`
`(__c_locale __cloc, const char * __s, size_t __refs = 0)`
`[explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.

s The name of a locale.

refs Refcount to pass to the base class.

Definition at line 49 of file messages_members.h.

5.552.3.3 `template<typename _CharT > std::messages< _CharT`
`>::~~messages() [protected, virtual]`

Destructor.

Definition at line 78 of file messages_members.h.

5.552.4 Member Data Documentation

5.552.4.1 `template<typename _CharT > locale::id std::messages< _CharT`
`>::id [static]`

Numpunct facet id.

Definition at line 1708 of file locale_facets_nonio.h.

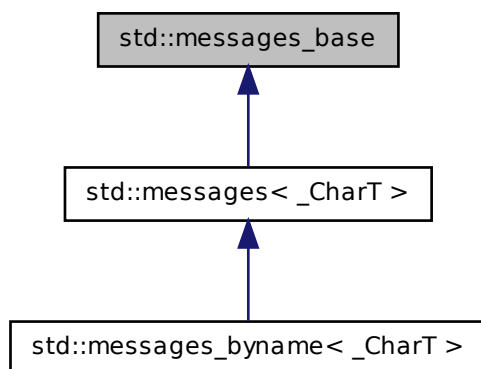
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [messages_members.h](#)

5.553 std::messages_base Struct Reference

Messages facet base class providing catalog typedef.

Inheritance diagram for std::messages_base:



Public Types

- typedef int **catalog**

5.553.1 Detailed Description

Messages facet base class providing catalog typedef.

Definition at line 1663 of file `locale_facets_nonio.h`.

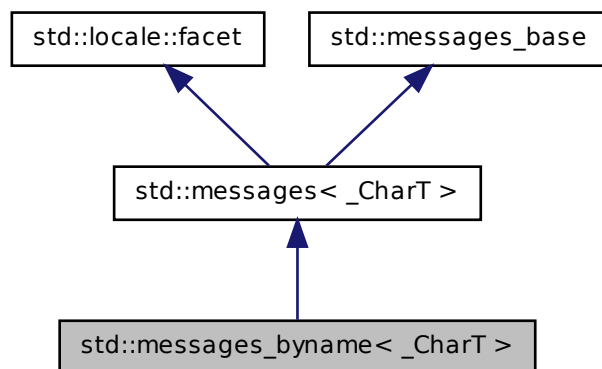
The documentation for this struct was generated from the following file:

- [locale_facets_nonio.h](#)

5.554 std::messages_byname< _CharT > Class Template Reference

class [messages_byname](#) [22.2.7.2].

Inheritance diagram for `std::messages_byname<_CharT>`:



Public Types

- typedef int **catalog**
- typedef `_CharT` **char_type**
- typedef **basic_string**< `_CharT` > **string_type**

Public Member Functions

- **messages_byname** (const char *__s, size_t __refs=0)
- void **close** (catalog __c) const
- **string_type** **get** (catalog __c, int __set, int __msgid, const **string_type** &__s) const
- catalog **open** (const **basic_string**< char > &, const **locale** &, const char *) const
- catalog **open** (const **basic_string**< char > &__s, const **locale** &__loc) const

Static Public Attributes

- static **locale::id** **id**

Protected Member Functions

- [string_type](#) **_M_convert_from_char** (char *) const
- char * **_M_convert_to_char** (const [string_type](#) &__msg) const
- virtual void **do_close** (catalog) const
- template<>
[wstring](#) **do_get** (catalog, int, int, const [wstring](#) &) const
- template<>
[string](#) **do_get** (catalog, int, int, const [string](#) &) const
- virtual [string_type](#) **do_get** (catalog, int, int, const [string_type](#) &__default) const
- virtual catalog **do_open** (const [basic_string](#)< char > &, const [locale](#) &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char * **_M_name_messages**

Friends

- class **locale::_Impl**

5.554.1 Detailed Description

template<typename _CharT> class std::messages_byname< _CharT >

class [messages_byname](#) [22.2.7.2].

Definition at line 1907 of file locale_facets_nonio.h.

5.554.2 Member Typedef Documentation

5.554.2.1 `template<typename _CharT > typedef _CharT std::messages_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::messages< _CharT >](#).

Definition at line 1910 of file `locale_facets_nonio.h`.

5.554.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::messages_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::messages< _CharT >](#).

Definition at line 1911 of file `locale_facets_nonio.h`.

5.554.3 Member Data Documentation

5.554.3.1 `template<typename _CharT > locale::id std::messages< _CharT >::id [static, inherited]`

Numpunct facet id.

Definition at line 1708 of file `locale_facets_nonio.h`.

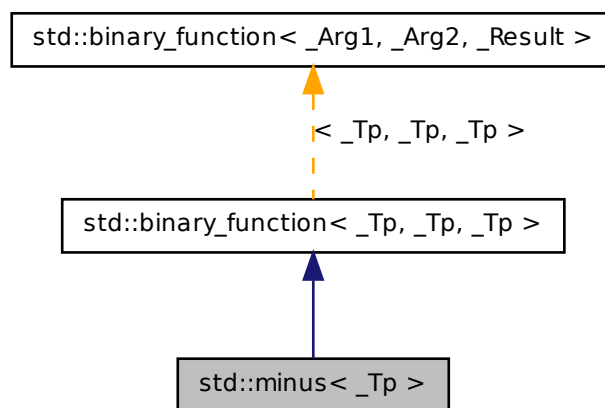
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [messages_members.h](#)

5.555 `std::minus< _Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::minus< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp` **operator()** (const `_Tp` &__x, const `_Tp` &__y) const

5.555.1 Detailed Description

`template<typename _Tp> struct std::minus< _Tp >`

One of the [math functors](#).

Definition at line 144 of file `stl_function.h`.

5.555.2 Member Typedef Documentation

5.555.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.555.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type` [inherited]

type of the return type

Definition at line 118 of file `stl_function.h`.

5.555.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

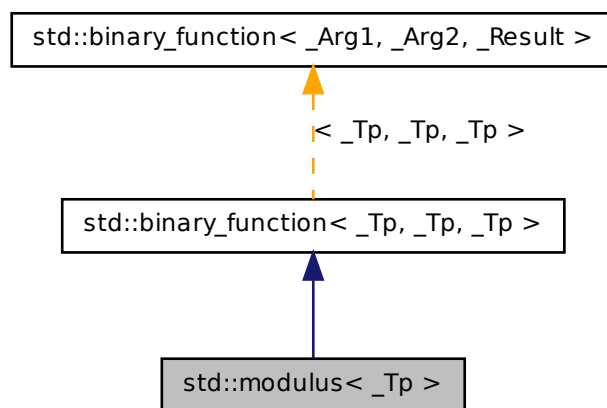
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.556 `std::modulus< _Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::modulus< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp` **operator()** (const `_Tp` &__x, const `_Tp` &__y) const

5.556.1 Detailed Description

`template<typename _Tp> struct std::modulus< _Tp >`

One of the [math functors](#).

Definition at line 171 of file `stl_function.h`.

5.556.2 Member Typedef Documentation

5.556.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.556.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.556.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

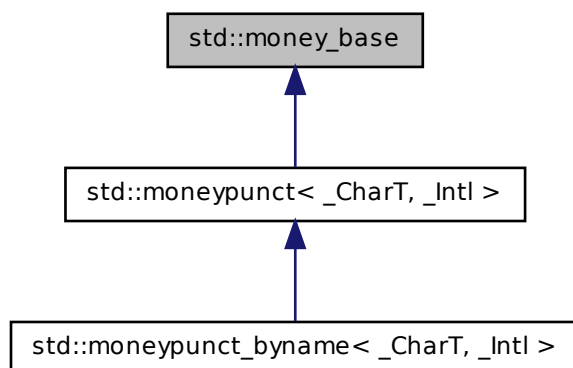
- [stl_function.h](#)

5.557 `std::money_base` Class Reference

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

Inheritance diagram for std::money_base:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- enum **part** {
 none, **space**, **symbol**, **sign**,
 value }

Static Public Member Functions

- static `_GLIBCXX_CONST` pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**

5.557.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

[moneypunct::pos_format\(\)](#) and [moneypunct::neg_format\(\)](#) for details of how these fields are interpreted.

Definition at line 835 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

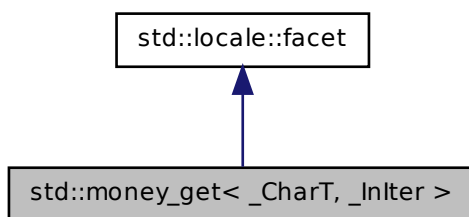
- [locale_facets_nonio.h](#)

5.558 `std::money_get< _CharT, _InIter >` Class Template Reference

Primary class template [money_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string.

Inheritance diagram for `std::money_get< _CharT, _InIter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_InIter` [iter_type](#)
- typedef [basic_string< _CharT >](#) [string_type](#)

Public Member Functions

- [money_get](#) (size_t __refs=0)
- [iter_type get](#) ([iter_type](#) __s, [iter_type](#) __end, bool __intl, [ios_base](#) &__io, [ios_base::iostate](#) &__err, [string_type](#) &__digits) const
- [iter_type get](#) ([iter_type](#) __s, [iter_type](#) __end, bool __intl, [ios_base](#) &__io, [ios_base::iostate](#) &__err, long double &__units) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~money_get](#) ()
- template<bool _Intl>
[iter_type _M_extract](#) ([iter_type](#) __s, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, [string](#) &__digits) const
- virtual [iter_type do_get](#) ([iter_type](#) __s, [iter_type](#) __end, bool __intl, [ios_base](#) &__io, [ios_base::iostate](#) &__err, [string_type](#) &__digits) const
- virtual [iter_type do_get](#) ([iter_type](#) __s, [iter_type](#) __end, bool __intl, [ios_base](#) &__io, [ios_base::iostate](#) &__err, long double &__units) const

Static Protected Member Functions

- static [__c_locale _S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const char *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale _S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale _S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.558.1 Detailed Description

template<typename _CharT, typename _InIter> class std::money_get< _CharT, _InIter >

Primary class template [money_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string. The [money_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [money_get](#) facet.

Definition at line 1365 of file locale_facets_nonio.h.

5.558.2 Member Typedef Documentation

**5.558.2.1 template<typename _CharT , typename _InIter > typedef _CharT
std::money_get< _CharT, _InIter >::char_type**

Public typedefs.

Definition at line 1371 of file locale_facets_nonio.h.

**5.558.2.2 template<typename _CharT , typename _InIter > typedef _InIter
std::money_get< _CharT, _InIter >::iter_type**

Public typedefs.

Definition at line 1372 of file locale_facets_nonio.h.

**5.558.2.3 template<typename _CharT , typename _InIter > typedef
basic_string<_CharT> std::money_get< _CharT, _InIter
>::string_type**

Public typedefs.

Definition at line 1373 of file locale_facets_nonio.h.

5.558.3 Constructor & Destructor Documentation

5.558.3.1 `template<typename _CharT, typename _InIter > std::money_get< _CharT, _InIter >::money_get (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 1387 of file locale_facets_nonio.h.

5.558.3.2 `template<typename _CharT, typename _InIter > virtual std::money_get< _CharT, _InIter >::~~money_get () [inline, protected, virtual]`

Destructor.

Definition at line 1455 of file locale_facets_nonio.h.

5.558.4 Member Function Documentation

5.558.4.1 `template<typename _CharT, typename _InIter > _InIter std::money_get< _CharT, _InIter >::do_get (iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units) const [protected, virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

Definition at line 363 of file locale_facets_nonio.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`.

Referenced by `std::money_get< _CharT, _InIter >::get()`.

5.558.4.2 `template<typename _CharT, typename _InIter > _InIter
std::money_get< _CharT, _InIter >::do_get (iter_type __s,
iter_type __end, bool __intl, ios_base & __io, ios_base::iostate &
__err, string_type & __digits) const [protected, virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

Definition at line 376 of file locale_facets_nonio.tcc.

References `std::ios_base::M_getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

5.558.4.3 `template<typename _CharT, typename _InIter > iter_type
std::money_get< _CharT, _InIter >::get (iter_type __s, iter_type
__end, bool __intl, ios_base & __io, ios_base::iostate & __err,
string_type & __digits) const [inline]`

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of [do_get\(\)](#).

Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use `_facet<moneypunct<CharT,intl> >`.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- digits* Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1448 of file locale_facets_nonio.h.

References `std::money_get< _CharT, _InIter >::do_get()`.

5.558.4.4 `template<typename _CharT, typename _InIter > iter_type
std::money_get< _CharT, _InIter >::get (iter_type __s, iter_type
__end, bool __intl, ios_base & __io, ios_base::iostate & __err, long
double & __units) const [inline]`

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *units* as an integral value `moneypunct::frac_digits()` * the actual amount. For example, the string \$10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. *units* is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- units* Place to store result of parsing.

Returns

- Iterator referencing first character beyond valid money amount.

Definition at line 1417 of file locale_facets_nonio.h.

References `std::money_get< _CharT, _InIter >::do_get()`.

5.558.5 Member Data Documentation

5.558.5.1 `template<typename _CharT, typename _InIter > locale::id
std::money_get< _CharT, _InIter >::id [static]`

Numpunct facet id.

Definition at line 1377 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

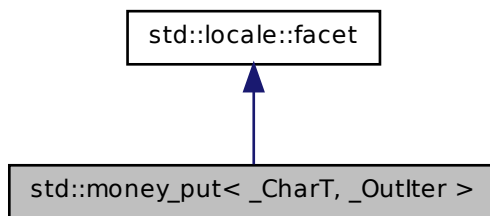
- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.559 `std::money_put< _CharT, _OutIter >` Class Template Reference

Primary class template [money_put](#).

This facet encapsulates the code to format and output a monetary amount.

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_OutIter` [iter_type](#)
- typedef [basic_string< _CharT >](#) [string_type](#)

Public Member Functions

- [money_put](#) (size_t __refs=0)
- [iter_type put](#) ([iter_type](#) __s, bool __intl, [ios_base](#) &__io, [char_type](#) __fill, const [string_type](#) &__digits) const
- [iter_type put](#) ([iter_type](#) __s, bool __intl, [ios_base](#) &__io, [char_type](#) __fill, long double __units) const

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~money_put()`
- `template<bool _Intl>`
`iter_type _M_insert(iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const`
- virtual `iter_type do_put(iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const`
- virtual `iter_type do_put(iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale(__c_locale &__cloc) throw()`
- static `void _S_create_c_locale(__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale(__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale()`
- static `_GLIBCXX_CONST const char * _S_get_c_name() throw()`
- static `__c_locale _S_lc_ctype_c_locale(__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.559.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::money_put<_CharT, _OutIter>`

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount. The `money_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_put` facet.

Definition at line 1516 of file `locale_facets_nonio.h`.

5.559.2 Member Typedef Documentation

5.559.2.1 `template<typename _CharT, typename _OutIter > typedef _CharT std::money_put< _CharT, _OutIter >::char_type`

Public typedefs.

Definition at line 1521 of file locale_facets_nonio.h.

5.559.2.2 `template<typename _CharT, typename _OutIter > typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 1522 of file locale_facets_nonio.h.

5.559.2.3 `template<typename _CharT, typename _OutIter > typedef basic_string<_CharT> std::money_put< _CharT, _OutIter >::string_type`

Public typedefs.

Definition at line 1523 of file locale_facets_nonio.h.

5.559.3 Constructor & Destructor Documentation

5.559.3.1 `template<typename _CharT, typename _OutIter > std::money_put< _CharT, _OutIter >::money_put (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 1537 of file locale_facets_nonio.h.

5.559.3.2 `template<typename _CharT, typename _OutIter > virtual std::money_put< _CharT, _OutIter >::~~money_put () [inline, protected, virtual]`

Destructor.

Definition at line 1587 of file locale_facets_nonio.h.

5.559.4 Member Function Documentation

5.559.4.1 `template<typename _CharT, typename _OutIter> _OutIter
std::money_put<_CharT, _OutIter>::do_put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, long double __units)
const [protected, virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#).

Parameters

s The stream to write to.

intl Parameter to use_facet<moneypunct<CharT,intl>>.

io Source of facets and io state.

fill char_type to use for padding.

units Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 568 of file locale_facets_nonio.tcc.

References `std::ios_base::getloc()`, and `std::__ctype_abstract_base<_CharT>::widen()`.

Referenced by `std::money_put<_CharT, _OutIter>::put()`.

5.559.4.2 `template<typename _CharT, typename _OutIter> _OutIter
std::money_put<_CharT, _OutIter>::do_put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, const string_type &
__digits) const [protected, virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#).

Parameters

- s* The stream to write to.
- intl* Parameter to use `_facet<moneypunct<CharT,intl> >`.
- io* Source of facets and io state.
- fill* `char_type` to use for padding.
- units* Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 606 of file `locale_facets_nonio.tcc`.

5.559.4.3 `template<typename _CharT , typename _OutIter > iter_type
std::money_put< _CharT, _OutIter >::put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, const string_type &
__digits) const [inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function works by returning the result of [do_put\(\)](#).

Parameters

- s* The stream to write to.
- intl* Parameter to use `_facet<moneypunct<CharT,intl> >`.
- io* Source of facets and io state.
- fill* `char_type` to use for padding.
- units* Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1580 of file locale_facets_nonio.h.

References std::money_put<_CharT, _OutIter>::do_put().

5.559.4.4 `template<typename _CharT, typename _OutIter> iter_type
std::money_put<_CharT, _OutIter>::put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, long double __units)
const [inline]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function works by returning the result of [do_put\(\)](#).

Parameters

s The stream to write to.

intl Parameter to use_facet<moneypunct<CharT,intl>>.

io Source of facets and io state.

fill char_type to use for padding.

units Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1557 of file locale_facets_nonio.h.

References std::money_put<_CharT, _OutIter>::do_put().

5.559.5 Member Data Documentation

5.559.5.1 `template<typename _CharT, typename _OutIter> locale::id
std::money_put<_CharT, _OutIter>::id [static]`

Numpunct facet id.

Definition at line 1527 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

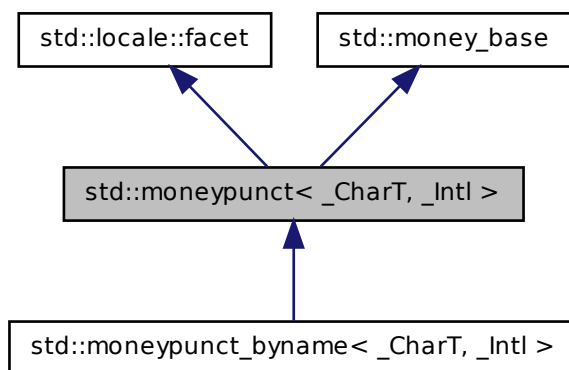
- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.560 `std::moneypunct< _CharT, _Intl >` Class Template Reference

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Inheritance diagram for `std::moneypunct< _CharT, _Intl >`:



Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- typedef `__moneypunct_cache< _CharT, _Intl >` `__cache_type`
- enum **part** {
`none`, `space`, `symbol`, `sign`,
`value` }
- typedef `_CharT` `char_type`
- typedef `basic_string< _CharT >` `string_type`

Public Member Functions

- [moneypunct](#) (size_t __refs=0)
 - [moneypunct](#) (__cache_type *__cache, size_t __refs=0)
 - [moneypunct](#) (__c_locale __cloc, const char *__s, size_t __refs=0)
 - [string_type curr_symbol](#) () const
 - [char_type decimal_point](#) () const
 - [int frac_digits](#) () const
 - [string grouping](#) () const
 - [string_type negative_sign](#) () const
 - [string_type positive_sign](#) () const
 - [char_type thousands_sep](#) () const
-
- pattern [pos_format](#) () const
 - pattern [neg_format](#) () const

Static Public Member Functions

- static _GLIBCXX_CONST pattern [_S_construct_pattern](#) (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * [_S_atoms](#)
- static const pattern [_S_default_pattern](#)
- static [locale::id](#) [id](#)
- static const bool [intl](#)

Protected Member Functions

- virtual [~moneypunct](#) ()
 - template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
 - void [_M_initialize_moneypunct](#) (__c_locale __cloc=0, const char *__name=0)
-
- template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
 - template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
 - template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
 - virtual [string_type do_curr_symbol](#) () const

- virtual [char_type do_decimal_point](#) () const
- virtual int [do_frac_digits](#) () const
- virtual [string do_grouping](#) () const
- virtual pattern [do_neg_format](#) () const
- virtual [string_type do_negative_sign](#) () const
- virtual pattern [do_pos_format](#) () const
- virtual [string_type do_positive_sign](#) () const
- virtual [char_type do_thousands_sep](#) () const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.560.1 Detailed Description

`template<typename _CharT, bool _Intl> class std::moneypunct< _CharT, _Intl >`

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 929 of file `locale_facets_nonio.h`.

5.560.2 Member Typedef Documentation

5.560.2.1 `template<typename _CharT, bool _Intl> typedef _CharT std::moneypunct< _CharT, _Intl >::char_type`

Public typedefs.

Reimplemented in `std::moneypunct_byname< _CharT, _Intl >`.

Definition at line 935 of file `locale_facets_nonio.h`.

```
5.560.2.2  template<typename _CharT, bool _Intl> typedef
            basic_string<_CharT> std::moneypunct< _CharT, _Intl
            >::string_type
```

Public typedefs.

Reimplemented in [std::moneypunct_byname< _CharT, _Intl >](#).

Definition at line 936 of file locale_facets_nonio.h.

5.560.3 Constructor & Destructor Documentation

```
5.560.3.1  template<typename _CharT, bool _Intl> std::moneypunct< _CharT,
            _Intl >::moneypunct ( size_t __refs = 0 ) [inline, explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 958 of file locale_facets_nonio.h.

```
5.560.3.2  template<typename _CharT, bool _Intl> std::moneypunct< _CharT,
            _Intl >::moneypunct ( __cache_type * __cache, size_t __refs = 0 )
            [inline, explicit]
```

Constructor performs initialization.

This is an internal constructor.

Parameters

cache Cache for optimization.

refs Passed to the base facet class.

Definition at line 971 of file locale_facets_nonio.h.

```
5.560.3.3  template<typename _CharT, bool _Intl> std::moneypunct< _CharT,
            _Intl >::moneypunct ( __c_locale __cloc, const char * __s, size_t
            __refs = 0 ) [inline, explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.
s The name of a locale.
refs Passed to the base facet class.

Definition at line 986 of file locale_facets_nonio.h.

5.560.3.4 `template<typename _CharT, bool _Intl> virtual std::moneypunct<_CharT, _Intl >::~~moneypunct () [protected, virtual]`

Destructor.

5.560.4 Member Function Documentation

5.560.4.1 `template<typename _CharT, bool _Intl> string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning `moneypunct<char_type>::do_curr_symbol()`.

Returns

string_type representing a currency symbol.

Definition at line 1056 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

5.560.4.2 `template<typename _CharT, bool _Intl> char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `moneypunct<char_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1000 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

5.560.4.3 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_curr_symbol () const
[inline, protected, virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

[curr_symbol\(\)](#) for details.

Returns

string_type representing a currency symbol.

Definition at line 1202 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::curr_symbol()`.

5.560.4.4 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct< _CharT, _Intl >::do_decimal_point () const
[inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1164 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::decimal_point()`.

5.560.4.5 `template<typename _CharT, bool _Intl> virtual int
std::moneypunct< _CharT, _Intl >::do_frac_digits () const
[inline, protected, virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

[frac_digits\(\)](#) for details.

Returns

Number of digits in amount fraction.

Definition at line 1242 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::frac_digits()`.

5.560.4.6 `template<typename _CharT, bool _Intl> virtual string
std::moneypunct<_CharT, _Intl >::do_grouping () const
[inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1189 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::grouping()`.

5.560.4.7 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct<_CharT, _Intl >::do_neg_format () const
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

[neg_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1270 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::neg_format()`.

5.560.4.8 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_negative_sign () const
[inline, protected, virtual]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

[negative_sign\(\)](#) for details.

Returns

string_type representing a negative sign.

Definition at line 1228 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::negative_sign()`.

5.560.4.9 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct< _CharT, _Intl >::do_pos_format () const
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

[pos_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1256 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.

5.560.4.10 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_positive_sign () const
[inline, protected, virtual]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

[positive_sign\(\)](#) for details.

Returns

string_type representing a positive sign.

Definition at line 1215 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::positive_sign()`.

5.560.4.11 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct<_CharT, _Intl >::do_thousands_sep () const
[inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1176 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::thousands_sep()`.

5.560.4.12 `template<typename _CharT, bool _Intl> int std::moneypunct<
_CharT, _Intl >::frac_digits () const [inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

Returns

Number of digits in amount fraction.

Definition at line 1106 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

5.560.4.13 `template<typename _CharT, bool _Intl> string std::moneypunct< _CharT, _Intl >::grouping () const [inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `and` is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1043 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_grouping()`.

5.560.4.14 `template<typename _CharT, bool _Intl> pattern std::moneypunct< _CharT, _Intl >::neg_format () const [inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none},

`curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1146 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_neg_format()`.

5.560.4.15 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl>::negative_sign () const
[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1090 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_negative_sign()`.

5.560.4.16 `template<typename _CharT, bool _Intl> pattern std::moneypunct<
_CharT, _Intl>::pos_format () const [inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be

present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and [pos_format\(\)](#) pattern {symbol,sign,value,none}, [curr_symbol\(\)](#) == '\$' [positive_sign\(\)](#) == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1142 of file locale_facets_nonio.h.

References [std::moneypunct< _CharT, _Intl >::do_pos_format\(\)](#).

5.560.4.17 `template<typename _CharT, bool _Intl> string_type
std::moneypunct< _CharT, _Intl >::positive_sign () const
[inline]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning [moneypunct<char_type>::do_positive_sign\(\)](#).

If the return value contains more than one character, the first character appears in the position indicated by [pos_format\(\)](#) and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1073 of file locale_facets_nonio.h.

References [std::moneypunct< _CharT, _Intl >::do_positive_sign\(\)](#).

5.560.4.18 `template<typename _CharT, bool _Intl> char_type
std::moneypunct< _CharT, _Intl >::thousands_sep () const
[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning [moneypunct<char_type>::do_thousands_sep\(\)](#).

Returns

`char_type` representing a thousands separator.

Definition at line 1013 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_thousands_sep()`.

5.560.5 Member Data Documentation

5.560.5.1 `template<typename _CharT, bool _Intl> locale::id
std::moneypunct<_CharT, _Intl >::id [static]`

Numpunct facet id.

Definition at line 948 of file locale_facets_nonio.h.

5.560.5.2 `template<typename _CharT, bool _Intl> const bool
std::moneypunct<_CharT, _Intl >::intl [static]`

This value is provided by the standard, but no reason for its /// existence.

Reimplemented in [std::moneypunct_byname<_CharT, _Intl >](#).

Definition at line 946 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

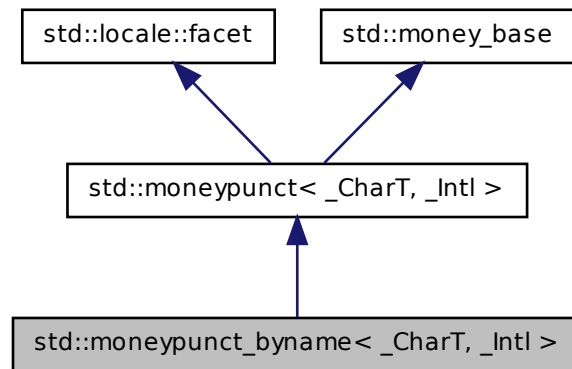
- [locale_facets_nonio.h](#)

5.561 `std::moneypunct_byname<_CharT, _Intl >` Class Template Reference

class [moneypunct_byname](#) [22.2.6.4].

5.561 `std::moneypunct_byname<_CharT, _Intl >` Class Template Reference 2737

Inheritance diagram for `std::moneypunct_byname<_CharT, _Intl >`:



Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- typedef `__moneypunct_cache<_CharT, _Intl >` `__cache_type`
- typedef `_CharT` `char_type`
- enum `part` {
 `none`, `space`, `symbol`, `sign`,
 `value` }
- typedef `basic_string<_CharT >` `string_type`

Public Member Functions

- `moneypunct_byname` (const char *__s, size_t __refs=0)
- `string_type curr_symbol` () const
- `char_type decimal_point` () const
- int `frac_digits` () const
- `string_type grouping` () const
- `string_type negative_sign` () const
- `string_type positive_sign` () const
- `char_type thousands_sep` () const

- pattern [pos_format](#) () const
- pattern [neg_format](#) () const

Static Public Member Functions

- static `_GLIBCXX_CONST` pattern **`_S_construct_pattern`** (char `__precedes`, char `__space`, char `__posn`) throw ()

Static Public Attributes

- static const char * **`_S_atoms`**
- static const pattern **`_S_default_pattern`**
- static [locale::id](#) **`id`**
- static const bool **`intl`**

Protected Member Functions

- template<>
void **`_M_initialize_moneypunct`** (`__c_locale`, const char *)
- void **`_M_initialize_moneypunct`** (`__c_locale` `__cloc`=0, const char * `__name`=0)
- template<>
void **`_M_initialize_moneypunct`** (`__c_locale`, const char *)
- template<>
void **`_M_initialize_moneypunct`** (`__c_locale`, const char *)
- template<>
void **`_M_initialize_moneypunct`** (`__c_locale`, const char *)
- virtual [string_type](#) **`do_curr_symbol`** () const
- virtual [char_type](#) **`do_decimal_point`** () const
- virtual int **`do_frac_digits`** () const
- virtual [string](#) **`do_grouping`** () const
- virtual pattern **`do_neg_format`** () const
- virtual [string_type](#) **`do_negative_sign`** () const
- virtual pattern **`do_pos_format`** () const
- virtual [string_type](#) **`do_positive_sign`** () const
- virtual [char_type](#) **`do_thousands_sep`** () const

Static Protected Member Functions

- static `__c_locale` **`_S_clone_c_locale`** (`__c_locale` & `__cloc`) throw ()
- static void **`_S_create_c_locale`** (`__c_locale` & `__cloc`, const char * `__s`, `__c_locale` `__old`=0)

5.561 `std::moneypunct_byname<_CharT, _Intl>` Class Template Reference **2739**

- static void `_S_destroy_c_locale` (`__c_locale &__cloc`)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST const char * _S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc, const char *__s`)

Friends

- class `locale::_Impl`

5.561.1 Detailed Description

`template<typename _CharT, bool _Intl> class std::moneypunct_byname< _CharT, _Intl >`

class [moneypunct_byname](#) [22.2.6.4].

Definition at line 1319 of file `locale_facets_nonio.h`.

5.561.2 Member Typedef Documentation

5.561.2.1 `template<typename _CharT, bool _Intl> typedef _CharT
std::moneypunct_byname< _CharT, _Intl >::char_type`

Public typedefs.

Reimplemented from [std::moneypunct< _CharT, _Intl >](#).

Definition at line 1322 of file `locale_facets_nonio.h`.

5.561.2.2 `template<typename _CharT, bool _Intl> typedef
basic_string<_CharT> std::moneypunct_byname< _CharT, _Intl
>::string_type`

Public typedefs.

Reimplemented from [std::moneypunct< _CharT, _Intl >](#).

Definition at line 1323 of file `locale_facets_nonio.h`.

5.561.3 Member Function Documentation

5.561.3.1 `template<typename _CharT, bool _Intl> string_type
std::moneypunct< _CharT, _Intl >::curr_symbol () const
[inline, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning [moneypunct<char_type>::do_curr_symbol\(\)](#).

Returns

string_type representing a currency symbol.

Definition at line 1056 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

5.561.3.2 `template<typename _CharT, bool _Intl> char_type
std::moneypunct< _CharT, _Intl >::decimal_point () const
[inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning [moneypunct<char_type>::do_decimal_point\(\)](#).

Returns

char_type representing a decimal point.

Definition at line 1000 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

5.561.3.3 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_curr_symbol () const
[inline, protected, virtual, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

[curr_symbol\(\)](#) for details.

5.561 std::moneypunct_byname<_CharT, _Intl> Class Template Reference2741

Returns

string_type representing a currency symbol.

Definition at line 1202 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::curr_symbol().

5.561.3.4 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct<_CharT, _Intl>::do_decimal_point () const
[inline, protected, virtual, inherited]`

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1164 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::decimal_point().

5.561.3.5 `template<typename _CharT, bool _Intl> virtual int
std::moneypunct<_CharT, _Intl>::do_frac_digits () const
[inline, protected, virtual, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

[frac_digits\(\)](#) for details.

Returns

Number of digits in amount fraction.

Definition at line 1242 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::frac_digits().

5.561.3.6 `template<typename _CharT, bool _Intl> virtual string
std::moneypunct< _CharT, _Intl >::do_grouping () const
[inline, protected, virtual, inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1189 of file locale_facets_nonio.h.

Referenced by `std::moneypunct< _CharT, _Intl >::grouping()`.

5.561.3.7 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct< _CharT, _Intl >::do_neg_format () const
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

[neg_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1270 of file locale_facets_nonio.h.

Referenced by `std::moneypunct< _CharT, _Intl >::neg_format()`.

5.561.3.8 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_negative_sign () const
[inline, protected, virtual, inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

5.561 std::moneypunct_byname<_CharT, _Intl > Class Template Reference2743

See also

[negative_sign\(\)](#) for details.

Returns

string_type representing a negative sign.

Definition at line 1228 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl >::negative_sign().

5.561.3.9 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct<_CharT, _Intl >::do_pos_format () const
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

[pos_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1256 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl >::pos_format().

5.561.3.10 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct<_CharT, _Intl >::do_positive_sign () const
[inline, protected, virtual, inherited]`

Return positive sign string.

This function returns a *string_type* to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

[positive_sign\(\)](#) for details.

Returns

string_type representing a positive sign.

Definition at line 1215 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::positive_sign()`.

5.561.3.11 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct<_CharT, _Intl >::do_thousands_sep () const
[inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1176 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::thousands_sep()`.

5.561.3.12 `template<typename _CharT, bool _Intl> int std::moneypunct<
_CharT, _Intl >::frac_digits () const [inline, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

Returns

Number of digits in amount fraction.

Definition at line 1106 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

5.561.3.13 `template<typename _CharT, bool _Intl> string std::moneypunct<
_CharT, _Intl >::grouping () const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

5.561 std::moneypunct_byname<_CharT, _Intl> Class Template Reference 2745

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the [grouping\(\)](#) returns and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling [moneypunct<char_type>::do_grouping\(\)](#).

Returns

string representing grouping specification.

Definition at line 1043 of file locale_facets_nonio.h.

References [std::moneypunct<_CharT, _Intl>::do_grouping\(\)](#).

5.561.3.14 template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format() const [inline, inherited]

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning [moneypunct<char_type>::do_pos_format\(\)](#) or [moneypunct<char_type>::do_neg_format\(\)](#).

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of [curr_symbol\(\)](#) may be present. The sign field indicates that the value of [positive_sign\(\)](#) or [negative_sign\(\)](#) must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and [pos_format\(\)](#) pattern {symbol,sign,value,none}, [curr_symbol\(\)](#) == '\$' [positive_sign\(\)](#) == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1146 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_neg_format()`.

5.561.3.15 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl >::negative_sign () const
[inline, inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1090 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_negative_sign()`.

5.561.3.16 `template<typename _CharT, bool _Intl> pattern std::moneypunct<
_CharT, _Intl >::pos_format () const [inline, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

5.561 std::moneypunct_byname<_CharT, _Intl> Class Template Reference2747

Returns

Pattern for money values.

Definition at line 1142 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_pos_format()`.

5.561.3.17 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl>::positive_sign () const
[inline, inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1073 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_positive_sign()`.

5.561.3.18 `template<typename _CharT, bool _Intl> char_type
std::moneypunct<_CharT, _Intl>::thousands_sep () const
[inline, inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1013 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_thousands_sep()`.

5.561.4 Member Data Documentation

5.561.4.1 `template<typename _CharT, bool _Intl> locale::id`
`std::moneypunct<_CharT, _Intl >::id` [**static**, **inherited**]

Numpunct facet id.

Definition at line 948 of file `locale_facets_nonio.h`.

5.561.4.2 `template<typename _CharT, bool _Intl> const bool`
`std::moneypunct_byname<_CharT, _Intl >::intl` [**static**]

This value is provided by the standard, but no reason for its /// existence.

Reimplemented from `std::moneypunct<_CharT, _Intl >`.

Definition at line 1325 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

5.562 `std::move_iterator<_Iterator >` Class Template Reference

Public Types

- `typedef __traits_type::difference_type` **difference_type**
- `typedef __traits_type::iterator_category` **iterator_category**
- `typedef _Iterator` **iterator_type**
- `typedef _Iterator` **pointer**
- `typedef value_type &&` **reference**
- `typedef __traits_type::value_type` **value_type**

Public Member Functions

- **move_iterator** (iterator_type __i)
- `template<typename _Iter >`
move_iterator (const [move_iterator](#)<_Iter > &__i)
- iterator_type **base** () const
- reference **operator*** () const
- [move_iterator](#) **operator+** (difference_type __n) const
- [move_iterator](#) **operator++** (int)

5.563 `std::multimap<_Key, _Tp, _Compare, _Alloc>` Class Template Reference 2749

- [move_iterator](#) & `operator++()`
- [move_iterator](#) & `operator+= (difference_type __n)`
- [move_iterator](#) `operator-` (difference_type __n) const
- [move_iterator](#) & `operator--()`
- [move_iterator](#) `operator--` (int)
- [move_iterator](#) & `operator-= (difference_type __n)`
- pointer `operator->()` const
- reference `operator[]` (difference_type __n) const

Protected Types

- typedef [iterator_traits](#)<_Iterator> __traits_type

Protected Attributes

- _Iterator _M_current

5.562.1 Detailed Description

`template<typename _Iterator> class std::move_iterator<_Iterator>`

Class template [move_iterator](#) is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 894 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

5.563 `std::multimap<_Key, _Tp, _Compare, _Alloc>` Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Rep_type::const_iterator` **const_iterator**
- typedef `_Pair_alloc_type::const_pointer` **const_pointer**
- typedef `_Pair_alloc_type::const_reference` **const_reference**
- typedef `_Rep_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Rep_type::difference_type` **difference_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Pair_alloc_type::pointer` **pointer**
- typedef `_Pair_alloc_type::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse_iterator**
- typedef `_Rep_type::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- `multimap` ()
- `multimap` (const `_Compare` &__comp, const `allocator_type` &__a=allocator_type())
- `multimap` (`multimap` &&__x)
- `multimap` (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=__Compare(), const `allocator_type` &__a=allocator_type())
- `multimap` (const `multimap` &__x)
- template<typename `_InputIterator` >
`multimap` (`_InputIterator` __first, `_InputIterator` __last)
- template<typename `_InputIterator` >
`multimap` (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp, const `allocator_type` &__a=allocator_type())
- iterator `begin` ()
- const_iterator `begin` () const
- const_iterator `cbegin` () const
- const_iterator `cend` () const
- void `clear` ()
- size_type `count` (const `key_type` &__x) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- bool `empty` () const
- iterator `end` ()
- const_iterator `end` () const

- `std::pair< const_iterator, const_iterator > equal_range` (const key_type &__x) const
- `std::pair< iterator, iterator > equal_range` (const key_type &__x)
- iterator `erase` (iterator __first, iterator __last)
- size_type `erase` (const key_type &__x)
- iterator `erase` (iterator __position)
- iterator `find` (const key_type &__x)
- const_iterator `find` (const key_type &__x) const
- allocator_type `get_allocator` () const
- iterator `insert` (const value_type &__x)
- iterator `insert` (iterator __position, const value_type &__x)
- template<typename _InputIterator >
void `insert` (_InputIterator __first, _InputIterator __last)
- void `insert` (initializer_list< value_type > __l)
- key_compare `key_comp` () const
- const_iterator `lower_bound` (const key_type &__x) const
- iterator `lower_bound` (const key_type &__x)
- size_type `max_size` () const
- `multimap` & `operator=` (const `multimap` &__x)
- `multimap` & `operator=` (`multimap` &&__x)
- `multimap` & `operator=` (initializer_list< value_type > __l)
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () const
- size_type `size` () const
- void `swap` (`multimap` &__x)
- const_iterator `upper_bound` (const key_type &__x) const
- iterator `upper_bound` (const key_type &__x)
- value_compare `value_comp` () const

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool **operator**< (const `multimap`< _K1, _T1, _C1, _A1 > &, const `multimap`< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool **operator**== (const `multimap`< _K1, _T1, _C1, _A1 > &, const `multimap`< _K1, _T1, _C1, _A1 > &)

5.563.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> class
std::multimap< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 86 of file `stl_multimap.h`.

5.563.2 Constructor & Destructor Documentation

5.563.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap () [inline]`

Default constructor creates no elements.

Definition at line 148 of file `stl_multimap.h`.

5.563.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (const _Compare & __comp, const allocator_type & __a = allocator_type()) [inline, explicit]`

Creates a `multimap` with no elements.

Parameters

comp A comparison object.

a An allocator object.

Definition at line 157 of file `stl_multimap.h`.

5.563 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

5.563.2.3 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc
>::multimap (const multimap< _Key, _Tp, _Compare, _Alloc > &
__x) [inline]`

Multimap copy constructor.

Parameters

x A multimap of identical element and allocator types.

The newly-created multimap uses a copy of the allocation object used by *x*.

Definition at line 168 of file `stl_multimap.h`.

5.563.2.4 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc
>::multimap (multimap< _Key, _Tp, _Compare, _Alloc > && __x
) [inline]`

Multimap move constructor.

Parameters

x A multimap of identical element and allocator types.

The newly-created multimap contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multimap.

Definition at line 179 of file `stl_multimap.h`.

5.563.2.5 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc
>::multimap (initializer_list< value_type > __l, const _Compare
& __comp = _Compare(), const allocator_type & __a =
allocator_type()) [inline]`

Builds a multimap from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison functor.

a An allocator object.

Create a multimap consisting of copies of the elements from the [initializer_list](#). This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 192 of file `stl_multimap.h`.

```
5.563.2.6  template<typename _Key, typename _Tp, typename _Compare =
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
             _Key, _Tp>>> template<typename _InputIterator >
             std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
             _InputIterator __first, _InputIterator __last ) [inline]
```

Builds a multimap from a range.

Parameters

first An input iterator.

last An input iterator.

Create a multimap consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(first,last)`).

Definition at line 209 of file `stl_multimap.h`.

```
5.563.2.7  template<typename _Key, typename _Tp, typename _Compare =
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
             _Key, _Tp>>> template<typename _InputIterator >
             std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
             _InputIterator __first, _InputIterator __last, const _Compare &
             __comp, const allocator_type & __a = allocator_type() )
             [inline]
```

Builds a multimap from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a multimap consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(first,last)`).

Definition at line 225 of file `stl_multimap.h`.

5.563.3 Member Function Documentation

5.563.3.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () [inline]`

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 304 of file stl_multimap.h.

5.563.3.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 313 of file stl_multimap.h.

5.563.3.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 377 of file stl_multimap.h.

5.563.3.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 386 of file stl_multimap.h.

5.563.3.5 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::clear () [inline]`

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 601 of file `stl_multimap.h`.

5.563.3.6 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::count (const key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

Definition at line 658 of file `stl_multimap.h`.

5.563.3.7 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::crbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 395 of file `stl_multimap.h`.

5.563.3.8 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 404 of file `stl_multimap.h`.

5.563 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

5.563.3.9 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> bool std::multimap< _Key, _Tp, _Compare, _Alloc
>::empty () const [inline]`

Returns true if the multimap is empty.

Definition at line 411 of file stl_multimap.h.

5.563.3.10 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 331 of file stl_multimap.h.

5.563.3.11 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 322 of file stl_multimap.h.

5.563.3.12 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::pair<iterator, iterator> std::multimap< _Key,
_Tp, _Compare, _Alloc >::equal_range (const key_type & __x)
[inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 725 of file stl_multimap.h.

5.563.3.13 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> std::pair<const_iterator, const_iterator>
std::multimap<_Key, _Tp, _Compare, _Alloc>::equal_range(
const key_type & __x) const [inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 742 of file stl_multimap.h.

5.563.3.14 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::erase(iterator __position) [inline]`

Erases an element from a multimap.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, [end\(\)](#) is returned.

5.563 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 509 of file stl_multimap.h.

```
5.563.3.15 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare,  
_Alloc >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 539 of file stl_multimap.h.

```
5.563.3.16 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::erase ( iterator __first, iterator __last ) [inline]
```

Erases a [first,last) range of elements from a multimap.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 558 of file stl_multimap.h.

5.563.3.17 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::multimap<_Key, _Tp,
_Compare, _Alloc>::find (const key_type & __x) const
[inline]`

Tries to locate an element in a multimap.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 649 of file stl_multimap.h.

5.563.3.18 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::find (const key_type & __x) [inline]`

Tries to locate an element in a multimap.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 634 of file stl_multimap.h.

5.563 std::multimap<_Key, _Tp, _Compare, _Alloc> Class Template Reference

5.563.3.19 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> allocator_type std::multimap<_Key, _Tp,
_Compare, _Alloc>::get_allocator() const [inline]`

Get a copy of the memory allocation object.

Definition at line 294 of file stl_multimap.h.

5.563.3.20 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::insert(const value_type & __x) [inline]`

Inserts a [std::pair](#) into the multimap.

Parameters

x Pair to be inserted (see [std::make_pair](#) for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a [std::map](#) the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 438 of file stl_multimap.h.

5.563.3.21 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::insert(iterator __position, const value_type & __x)
[inline]`

Inserts a [std::pair](#) into the multimap.

Parameters

position An iterator that serves as a hint as to where the pair should be inserted.

x Pair to be inserted (see [std::make_pair](#) for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a [std::map](#) the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 462 of file stl_multimap.h.

```
5.563.3.22  template<typename _Key, typename _Tp, typename _Compare =
               std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
               _Key, _Tp>>> template<typename _InputIterator > void
               std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
               _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 476 of file stl_multimap.h.

```
5.563.3.23  template<typename _Key, typename _Tp, typename _Compare =
               std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
               _Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc
               >::insert ( initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of std::pairs into the multimap.

Parameters

list A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 488 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

5.563 std::multimap<_Key, _Tp, _Compare, _Alloc > Class Template Reference

5.563.3.24 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> key_compare std::multimap<_Key, _Tp,
_Compare, _Alloc >::key_comp () const [inline]`

Returns the key comparison object out of which the multimap was constructed.

Definition at line 610 of file stl_multimap.h.

5.563.3.25 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> const_iterator std::multimap<_Key, _Tp,
_Compare, _Alloc >::lower_bound (const key_type & __x) const
[inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to [end\(\)](#).

Definition at line 688 of file stl_multimap.h.

5.563.3.26 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc >::lower_bound (const key_type & __x) [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 673 of file `stl_multimap.h`.

5.563.3.27 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare,
_Alloc>::max_size () const [inline]`

Returns the maximum size of the multimap.

Definition at line 421 of file `stl_multimap.h`.

5.563.3.28 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare,
_Alloc>::operator=(multimap<_Key, _Tp, _Compare, _Alloc >
&& __x) [inline]`

Multimap move assignment operator.

Parameters

x A multimap of identical element and allocator types.

The contents of *x* are moved into this multimap (without copying). *x* is a valid, but unspecified multimap.

Definition at line 263 of file `stl_multimap.h`.

5.563.3.29 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare,
_Alloc>::operator=(const multimap<_Key, _Tp, _Compare,
_Alloc > & __x) [inline]`

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

x A multimap of identical element and allocator types.

5.563 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 248 of file stl_multimap.h.

```
5.563.3.30  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare,
_Alloc >::operator= ( initializer_list< value_type > __l )
[inline]
```

Multimap list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a multimap with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 284 of file stl_multimap.h.

```
5.563.3.31  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::rbegin ( ) [inline]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 340 of file stl_multimap.h.

```
5.563.3.32  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::rbegin ( ) const [inline]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 349 of file stl_multimap.h.

5.563.3.33 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 358 of file `stl_multimap.h`.

5.563.3.34 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 367 of file `stl_multimap.h`.

5.563.3.35 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare,
_Alloc >::size () const [inline]`

Returns the size of the multimap.

Definition at line 416 of file `stl_multimap.h`.

5.563.3.36 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc
>::swap (multimap< _Key, _Tp, _Compare, _Alloc > & __x)
[inline]`

Swaps data with another multimap.

Parameters

x A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

5.563 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

Definition at line 591 of file stl_multimap.h.

Referenced by std::swap().

```
5.563.3.37  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp,  
             _Compare, _Alloc >::upper_bound ( const key_type & __x ) const  
             [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 708 of file stl_multimap.h.

```
5.563.3.38  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,  
             _Alloc >::upper_bound ( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 698 of file stl_multimap.h.

```
5.563.3.39  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> value_compare std::multimap< _Key, _Tp,  
             _Compare, _Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

Definition at line 618 of file stl_multimap.h.

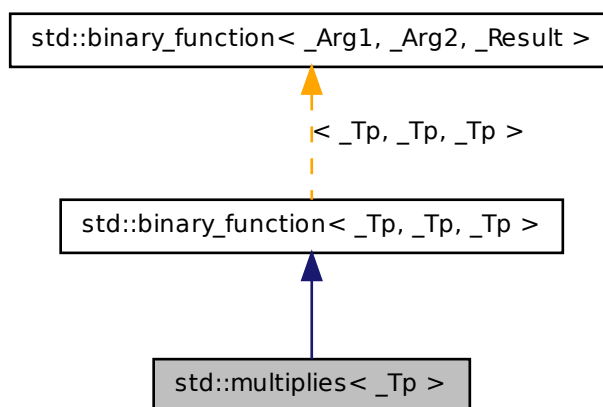
The documentation for this class was generated from the following file:

- [stl_multimap.h](#)

5.564 std::multiplies< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::multiplies< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

5.564.1 Detailed Description

template<typename _Tp> struct std::multiplies< _Tp >

One of the [math functors](#).

Definition at line 153 of file stl_function.h.

5.564.2 Member Typedef Documentation

**5.564.2.1 typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::first_argument_type [inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

**5.564.2.2 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type
[inherited]**

type of the return type

Definition at line 118 of file stl_function.h.

**5.564.2.3 typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::second_argument_type [inherited]**

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.565 std::multiset< _Key, _Compare, _Alloc > Class Template Reference

A standard container made up of elements, which can be retrieved in logarithmic time.

Public Types

- typedef _Alloc **allocator_type**

- typedef `_Rep_type::const_iterator` **const_iterator**
- typedef `_Key_alloc_type::const_pointer` **const_pointer**
- typedef `_Key_alloc_type::const_reference` **const_reference**
- typedef `_Rep_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Rep_type::difference_type` **difference_type**
- typedef `_Rep_type::const_iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Key_alloc_type::pointer` **pointer**
- typedef `_Key_alloc_type::reference` **reference**
- typedef `_Rep_type::const_reverse_iterator` **reverse_iterator**
- typedef `_Rep_type::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **multiset** ()
- **multiset** (const `_Compare` &__comp, const `allocator_type` &__a=allocator_type())
- template<typename `_InputIterator` >
multiset (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp, const `allocator_type` &__a=allocator_type())
- **multiset** (const **multiset** &__x)
- template<typename `_InputIterator` >
multiset (`_InputIterator` __first, `_InputIterator` __last)
- **multiset** (**multiset** &&__x)
- **multiset** (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=__Compare(), const `allocator_type` &__a=allocator_type())
- iterator **begin** () const
- iterator **cbegin** () const
- iterator **end** () const
- void **clear** ()
- `size_type` **count** (const `key_type` &__x) const
- `reverse_iterator` **crbegin** () const
- `reverse_iterator` **crend** () const
- bool **empty** () const
- iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- `size_type` **erase** (const `key_type` &__x)
- iterator **erase** (iterator __position)
- `allocator_type` **get_allocator** () const

- iterator [insert](#) (const value_type &__x)
 - iterator [insert](#) (iterator __position, const value_type &__x)
 - template<typename _InputIterator >
void [insert](#) (_InputIterator __first, _InputIterator __last)
 - void [insert](#) (initializer_list< value_type > __l)
 - key_compare [key_comp](#) () const
 - size_type [max_size](#) () const
 - multiset & [operator=](#) (const multiset &__x)
 - multiset & [operator=](#) (multiset &&__x)
 - multiset & [operator=](#) (initializer_list< value_type > __l)
 - reverse_iterator [rbegin](#) () const
 - reverse_iterator [rend](#) () const
 - size_type [size](#) () const
 - void [swap](#) (multiset &__x)
 - value_compare [value_comp](#) () const
-
- iterator [find](#) (const key_type &__x)
 - const_iterator [find](#) (const key_type &__x) const
-
- iterator [lower_bound](#) (const key_type &__x)
 - const_iterator [lower_bound](#) (const key_type &__x) const
-
- iterator [upper_bound](#) (const key_type &__x)
 - const_iterator [upper_bound](#) (const key_type &__x) const
-
- std::pair< iterator, iterator > [equal_range](#) (const key_type &__x)
 - std::pair< const_iterator, const_iterator > [equal_range](#) (const key_type &__x)
const
 - template<typename _K1, typename _C1, typename _A1 >
bool [operator==](#) (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)
 - template<typename _K1, typename _C1, typename _A1 >
bool [operator<](#) (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)

5.565.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::multiset< _Key, _Compare, _Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an

associative container (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 84 of file `stl_multiset.h`.

5.565.2 Constructor & Destructor Documentation

5.565.2.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset () [inline]`

Default constructor creates no elements.

Definition at line 129 of file `stl_multiset.h`.

5.565.2.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset (const _Compare & __comp, const
allocator_type & __a = allocator_type()) [inline,
explicit]`

Creates a multiset with no elements.

Parameters

comp Comparator to use.

a An allocator object.

Definition at line 138 of file `stl_multiset.h`.

5.565.2.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset
(_InputIterator __first, _InputIterator __last) [inline]`

Builds a multiset from a range.

Parameters

first An input iterator.

last An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 152 of file stl_multiset.h.

```
5.565.2.4 template<typename _Key, typename _Compare = std::less<_Key>,
    typename _Alloc = std::allocator<_Key>> template<typename
    _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset
    ( _InputIterator __first, _InputIterator __last, const _Compare &
    __comp, const allocator_type & __a = allocator_type() )
    [inline]
```

Builds a multiset from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 168 of file stl_multiset.h.

```
5.565.2.5 template<typename _Key, typename _Compare = std::less<_Key>,
    typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
    _Compare, _Alloc >::multiset ( const multiset< _Key, _Compare,
    _Alloc > & __x ) [inline]
```

Multiset copy constructor.

Parameters

x A multiset of identical element and allocator types.

The newly-created multiset uses a copy of the allocation object used by *x*.

Definition at line 181 of file stl_multiset.h.

5.565.2.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset (multiset< _Key, _Compare, _Alloc >
&& __x) [inline]`

Multiset move constructor.

Parameters

x A multiset of identical element and allocator types.

The newly-created multiset contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multiset.

Definition at line 192 of file `stl_multiset.h`.

5.565.2.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset (initializer_list< value_type > __l,
const _Compare & __comp = _Compare(), const allocator_type &
__a = allocator_type()) [inline]`

Builds a multiset from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison functor.

a An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 205 of file `stl_multiset.h`.

5.565.3 Member Function Documentation

5.565.3.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 285 of file `stl_multiset.h`.

5.565.3.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 322 of file stl_multiset.h.

5.565.3.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 331 of file stl_multiset.h.

5.565.3.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::multiset< _Key,
_Compare, _Alloc >::clear () [inline]`

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 541 of file stl_multiset.h.

5.565.3.5 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::count (const key_type & __x) const
[inline]`

Finds the number of elements with given key.

Parameters

x Key of elements to be located.

Returns

Number of elements with specified key.

Definition at line 552 of file stl_multiset.h.

5.565.3.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::crbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 340 of file `stl_multiset.h`.

5.565.3.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::crend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 349 of file `stl_multiset.h`.

5.565.3.8 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> bool std::multiset< _Key,
_Compare, _Alloc >::empty () const [inline]`

Returns true if the set is empty.

Definition at line 355 of file `stl_multiset.h`.

5.565.3.9 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 294 of file `stl_multiset.h`.

5.565.3.10 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::pair<iterator,
iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (
const key_type & __x) [inline]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 632 of file stl_multiset.h.

```
5.565.3.11  template<typename _Key, typename _Compare = std::less<_Key>,
                 typename _Alloc = std::allocator<_Key>> std::pair<const_iterator,
                 const_iterator> std::multiset< _Key, _Compare, _Alloc
                 >::equal_range ( const key_type & __x ) const  [inline]
```

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 636 of file stl_multiset.h.

```
5.565.3.12  template<typename _Key, typename _Compare = std::less<_Key>,
                 typename _Alloc = std::allocator<_Key>> iterator std::multiset<
                 _Key, _Compare, _Alloc >::erase ( iterator __position )
                 [inline]
```

Erases an element from a multiset.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 466 of file `stl_multiset.h`.

```
5.565.3.13  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> size_type std::multiset<
               _Key, _Compare, _Alloc >::erase ( const key_type & __x )
               [inline]
```

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 496 of file `stl_multiset.h`.

```
5.565.3.14  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> iterator std::multiset<
               _Key, _Compare, _Alloc >::erase ( iterator __first, iterator __last
               ) [inline]
```

Erases a [*first*,*last*) range of elements from a multiset.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 515 of file stl_multiset.h.

```
5.565.3.15 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::find ( const key_type & __x )
[inline]
```

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or [end\(\)](#) if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ([end\(\)](#)) iterator.

Definition at line 570 of file stl_multiset.h.

```
5.565.3.16 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator
std::multiset< _Key, _Compare, _Alloc >::find ( const key_type &
__x ) const [inline]
```

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or [end\(\)](#) if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ([end\(\)](#)) iterator.

Definition at line 574 of file `stl_multiset.h`.

```
5.565.3.17  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> allocator_type
              std::multiset< _Key, _Compare, _Alloc >::get_allocator ( ) const
              [inline]
```

Returns the memory allocation object.

Definition at line 276 of file `stl_multiset.h`.

```
5.565.3.18  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> iterator std::multiset<
              _Key, _Compare, _Alloc >::insert ( const value_type & __x )
              [inline]
```

Inserts an element into the multiset.

Parameters

x Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a [std::set](#) the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 396 of file `stl_multiset.h`.

```
5.565.3.19  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> iterator std::multiset<
              _Key, _Compare, _Alloc >::insert ( iterator __position, const
              value_type & __x ) [inline]
```

Inserts an element into the multiset.

Parameters

position An iterator that serves as a hint as to where the element should be inserted.

x Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a [std::set](#) the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 420 of file stl_multiset.h.

```
5.565.3.20 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > void std::multiset< _Key, _Compare, _Alloc
>::insert ( _InputIterator __first, _InputIterator __last )
[inline]
```

A template function that tries to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 433 of file stl_multiset.h.

```
5.565.3.21 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::multiset<
_Key, _Compare, _Alloc >::insert ( initializer_list< value_type >
__l ) [inline]
```

Attempts to insert a list of elements into the multiset.

Parameters

list A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 445 of file `stl_multiset.h`.

References `std::multiset<_Key, _Compare, _Alloc>::insert()`.

Referenced by `std::multiset<_Key, _Compare, _Alloc>::insert()`.

5.565.3.22 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> key_compare
std::multiset<_Key, _Compare, _Alloc>::key_comp () const
[inline]`

Returns the comparison object.

Definition at line 268 of file `stl_multiset.h`.

5.565.3.23 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc>::lower_bound (const key_type & __x)
[inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 591 of file `stl_multiset.h`.

5.565.3.24 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator
std::multiset<_Key, _Compare, _Alloc>::lower_bound (const
key_type & __x) const [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 595 of file stl_multiset.h.

5.565.3.25 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::max_size() const [inline]`

Returns the maximum size of the set.

Definition at line 365 of file stl_multiset.h.

5.565.3.26 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset<_Key, _Compare, _Alloc >::operator= (multiset<
_Key, _Compare, _Alloc > && __x) [inline]`

Multiset move assignment operator.

Parameters

x A multiset of identical element and allocator types.

The contents of *x* are moved into this multiset (without copying). *x* is a valid, but unspecified multiset.

Definition at line 235 of file stl_multiset.h.

5.565.3.27 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset<_Key, _Compare, _Alloc >::operator= (
initializer_list< value_type > __l) [inline]`

Multiset list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a multiset with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 256 of file stl_multiset.h.

5.565.3.28 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset< _Key, _Compare, _Alloc >::operator= (const
multiset< _Key, _Compare, _Alloc > & __x) [inline]`

Multiset assignment operator.

Parameters

x A multiset of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 220 of file stl_multiset.h.

5.565.3.29 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 303 of file stl_multiset.h.

5.565.3.30 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 312 of file stl_multiset.h.

5.565.3.31 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::size () const [inline]`

Returns the size of the set.

Definition at line 360 of file stl_multiset.h.

```
5.565.3.32  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> void std::multiset<
               _Key, _Compare, _Alloc >::swap ( multiset< _Key, _Compare,
               _Alloc > & __x ) [inline]
```

Swaps data with another multiset.

Parameters

x A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 380 of file stl_multiset.h.

Referenced by `std::swap()`.

```
5.565.3.33  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> iterator std::multiset<
               _Key, _Compare, _Alloc >::upper_bound ( const key_type & __x )
               [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 607 of file stl_multiset.h.

```
5.565.3.34  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> const_iterator
               std::multiset< _Key, _Compare, _Alloc >::upper_bound ( const
               key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 611 of file `stl_multiset.h`.

```
5.565.3.35  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> value_compare
              std::multiset<_Key, _Compare, _Alloc >::value_comp ( ) const
              [inline]
```

Returns the comparison object.

Definition at line 272 of file `stl_multiset.h`.

5.565.4 Friends And Related Function Documentation

```
5.565.4.1  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> template<typename _K1
              , typename _C1, typename _A1 > bool operator< ( const multiset<
              _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > & )
              [friend]
```

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

5.565.4.2 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> template<typename
 _K1, typename _C1, typename _A1 > bool operator==(const
 multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &
) [friend]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

The documentation for this class was generated from the following file:

- [stl_multiset.h](#)

5.566 std::mutex Class Reference

mutex

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **mutex** (const [mutex](#) &)
- void **lock** ()
- native_handle_type **native_handle** ()
- [mutex](#) & **operator=** (const [mutex](#) &)
- bool **try_lock** ()
- void **unlock** ()

5.566.1 Detailed Description

mutex

Definition at line 62 of file mutex.

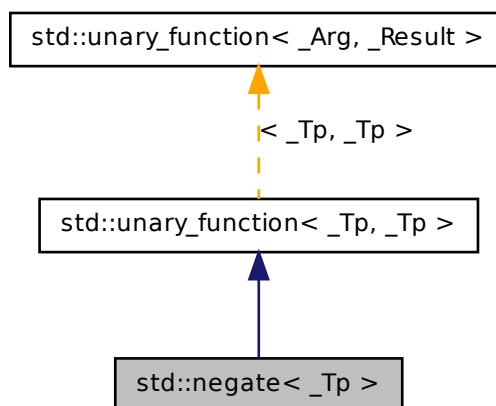
The documentation for this class was generated from the following file:

- [mutex](#)

5.567 std::negate< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::negate< _Tp >:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Tp` [result_type](#)

Public Member Functions

- `_Tp operator() (const _Tp &__x) const`

5.567.1 Detailed Description

`template<typename _Tp> struct std::negate< _Tp >`

One of the [math functors](#).

Definition at line 180 of file `stl_function.h`.

5.567.2 Member Typedef Documentation

5.567.2.1 `typedef _Tp std::unary_function< _Tp , _Tp >::argument_type`
`[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.567.2.2 `typedef _Tp std::unary_function< _Tp , _Tp >::result_type`
`[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.568 std::negative_binomial_distribution< _IntType > Class Template Reference

A [negative_binomial_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- **negative_binomial_distribution** (`_IntType __k=1`, `double __p=0.5`)
- **negative_binomial_distribution** (const [param_type](#) &__p)
- `_IntType k` () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`, const [param_type](#) &__p)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- `double p` () const
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
[std::basic_ostream](#)< `_CharT`, `_Traits` > & **operator<<** ([std::basic_ostream](#)< `_CharT`, `_Traits` > &, const [std::negative_binomial_distribution](#)< `_IntType1` > &)
- template<typename `_IntType1` >
bool **operator==** (const [std::negative_binomial_distribution](#)< `_IntType1` > &__d1, const [std::negative_binomial_distribution](#)< `_IntType1` > &__d2)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
[std::basic_istream](#)< `_CharT`, `_Traits` > & **operator>>** ([std::basic_istream](#)< `_CharT`, `_Traits` > &, [std::negative_binomial_distribution](#)< `_IntType1` > &)

5.568.1 Detailed Description

template<typename `_IntType = int`> class [std::negative_binomial_distribution](#)< `_IntType` >

A [negative_binomial_distribution](#) random number distribution. The formula for the negative binomial probability mass function is $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 3792 of file random.h.

5.568.2 Member Typedef Documentation

5.568.2.1 `template<typename _IntType = int> typedef _IntType
std::negative_binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3799 of file random.h.

5.568.3 Member Function Documentation

5.568.3.1 `template<typename _IntType = int> _IntType
std::negative_binomial_distribution< _IntType >::k () const
[inline]`

Return the k parameter of the distribution.

Definition at line 3848 of file random.h.

5.568.3.2 `template<typename _IntType = int> result_type
std::negative_binomial_distribution< _IntType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3884 of file random.h.

5.568.3.3 `template<typename _IntType = int> result_type
std::negative_binomial_distribution< _IntType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3877 of file random.h.

5.568.3.4 `template<typename _IntType > template<typename
UniformRandomNumberGenerator > negative
binomial_distribution< _IntType >::result_type
std::negative_binomial_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng)`

Generating functions.

Definition at line 1075 of file random.tcc.

5.568.3.5 `template<typename _IntType = int> double
std::negative_binomial_distribution< _IntType >::p () const
[inline]`

Return the p parameter of the distribution.

Definition at line 3855 of file random.h.

5.568.3.6 `template<typename _IntType = int> param_type
std::negative_binomial_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 3862 of file random.h.

5.568.3.7 `template<typename _IntType = int> void std::negative_binomial_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 3870 of file random.h.

5.568.3.8 `template<typename _IntType = int> void
std::negative_binomial_distribution< _IntType >::reset ()
[inline]`

Resets the distribution state.

Definition at line 3841 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

5.568.4 Friends And Related Function Documentation

5.568.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream<_CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > &
, const std::negative_binomial_distribution< _IntType1 > &)
[friend]`

Inserts a negative_binomial_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A negative_binomial_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.568.4.2 `template<typename _IntType = int> template<typename _IntType1
> bool operator== (const std::negative_binomial_distribution<
_IntType1 > & __d1, const std::negative_binomial_distribution<
_IntType1 > & __d2) [friend]`

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3906 of file random.h.

5.568.4.3 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream< _CharT, _Traits > & ,
std::negative_binomial_distribution< _IntType1 > &) [friend]`

Extracts a negative_binomial_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A negative_binomial_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.569 `std::negative_binomial_distribution< _IntType >::param_type` Struct Reference

Public Types

- typedef [negative_binomial_distribution](#)< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __k=1, double __p=0.5)
- _IntType **k** () const
- double **p** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.569.1 Detailed Description

```
template<typename _IntType = int> struct std::negative_binomial_ -  
distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3801 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.570 `std::nested_exception` Class Reference

Exception class with exception_ptr data member.

Inherited by `std::_Nested_exception< _Except >`.

Public Member Functions

- **nested_exception** (const [nested_exception](#) &)
- exception_ptr **nested_ptr** () const
- [nested_exception](#) & **operator=** (const [nested_exception](#) &)
- void **rethrow_nested** () const __attribute__((__noreturn__))

5.570.1 Detailed Description

Exception class with exception_ptr data member.

Definition at line 55 of file nested_exception.h.

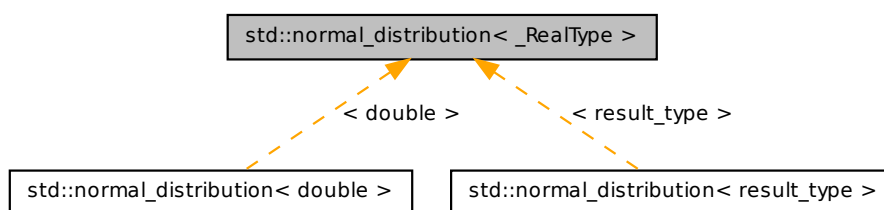
The documentation for this class was generated from the following file:

- [nested_exception.h](#)

5.571 std::normal_distribution< _RealType > Class Template Reference

A normal continuous distribution for random numbers.

Inheritance diagram for std::normal_distribution< _RealType >:



Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- [normal_distribution](#) ([result_type](#) __mean=[result_type](#)(0), [result_type](#) __stddev=[result_type](#)(1))
- **normal_distribution** (const [param_type](#) &__p)
- [result_type](#) max () const
- `_RealType` mean () const
- [result_type](#) min () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) operator() (`_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) operator() (`_UniformRandomNumberGenerator` &__urng)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) param () const
- void reset ()
- `_RealType` stddev () const

Friends

- template<typename `_RealType1`, typename `_CharT`, typename `_Traits` >
[std::basic_ostream](#)< `_CharT`, `_Traits` > & operator<< ([std::basic_ostream](#)< `_CharT`, `_Traits` > &, const [std::normal_distribution](#)< `_RealType1` > &)
- template<typename `_RealType1` >
bool operator== (const [std::normal_distribution](#)< `_RealType1` > &__d1, const [std::normal_distribution](#)< `_RealType1` > &__d2)
- template<typename `_RealType1`, typename `_CharT`, typename `_Traits` >
[std::basic_istream](#)< `_CharT`, `_Traits` > & operator>> ([std::basic_istream](#)< `_CharT`, `_Traits` > &, [std::normal_distribution](#)< `_RealType1` > &)

5.571.1 Detailed Description

template<typename `_RealType` = double> class [std::normal_distribution](#)< `_RealType` >

A normal continuous distribution for random numbers. The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 1990 of file random.h.

5.571.2 Member Typedef Documentation

5.571.2.1 template<typename _RealType = double> typedef _RealType std::normal_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 1997 of file random.h.

5.571.3 Constructor & Destructor Documentation

5.571.3.1 template<typename _RealType = double> std::normal_distribution< _RealType >::normal_distribution (result_type __mean = result_type (0), result_type __stddev = result_type (1)) [inline, explicit]

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2035 of file random.h.

5.571.4 Member Function Documentation

5.571.4.1 template<typename _RealType = double> result_type std::normal_distribution< _RealType >::max () const [inline]

Returns the least upper bound value of the distribution.

Definition at line 2092 of file random.h.

Referenced by std::normal_distribution< result_type >::max().

5.571.4.2 template<typename _RealType = double> _RealType std::normal_distribution< _RealType >::mean () const [inline]

Returns the mean of the distribution.

Definition at line 2056 of file random.h.

5.571.4.3 `template<typename _RealType = double> result_type
std::normal_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2085 of file random.h.

Referenced by `std::normal_distribution< result_type >::min()`.

5.571.4.4 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::normal_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2100 of file random.h.

Referenced by `std::normal_distribution< result_type >::operator()()`.

5.571.4.5 `template<typename _RealType > template<typename
_UniformRandomNumberGenerator > normal_distribution<
_RealType >::result_type std::normal_distribution< _RealType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1641 of file random.tcc.

References `std::log()`, and `std::sqrt()`.

5.571.4.6 `template<typename _RealType = double> void
std::normal_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2078 of file random.h.

5.571.4.7 `template<typename _RealType = double> param_type
std::normal_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2070 of file random.h.

Referenced by std::normal_distribution< result_type >::operator()().

5.571.4.8 `template<typename _RealType = double> void
std::normal_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2049 of file random.h.

Referenced by std::poisson_distribution< _IntType >::reset(), std::binomial_distribution< _IntType >::reset(), std::student_t_distribution< _RealType >::reset(), std::gamma_distribution< result_type >::reset(), and std::lognormal_distribution< _RealType >::reset().

5.571.4.9 `template<typename _RealType = double> _RealType
std::normal_distribution< _RealType >::stddev () const
[inline]`

Returns the standard deviation of the distribution.

Definition at line 2063 of file random.h.

5.571.5 Friends And Related Function Documentation

5.571.5.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::normal_distribution< _RealType1 > &) [friend]`

Inserts a normal_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A normal_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.571.5.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::normal_distribution<
_RealType1 > & __d1, const std::normal_distribution< _RealType1
> & __d2) [friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

5.571.5.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::normal_distribution< _RealType1 > &) [friend]`

Extracts a normal_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A normal_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.572 `std::normal_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [normal_distribution< _RealType >](#) `distribution_type`

Public Member Functions

- **param_type** (_RealType __mean=_RealType(0), _RealType __stddev=_RealType(1))
- _RealType **mean** () const
- _RealType **stddev** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.572.1 Detailed Description

template<typename _RealType = double> struct std::normal_distribution< _RealType >::param_type

Parameter type.

Definition at line 1999 of file random.h.

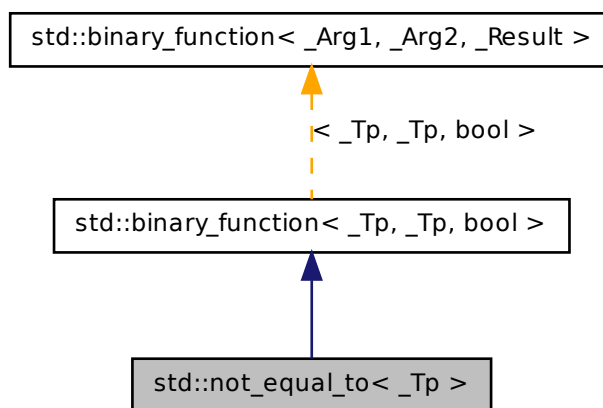
The documentation for this struct was generated from the following file:

- [random.h](#)

5.573 std::not_equal_to< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::not_equal_to< _Tp >`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.573.1 Detailed Description

`template<typename _Tp> struct std::not_equal_to< _Tp >`

One of the [comparison functors](#).

Definition at line 208 of file `stl_function.h`.

5.573.2 Member Typedef Documentation

5.573.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.573.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.573.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

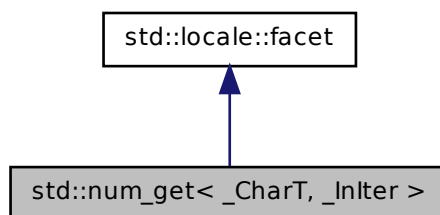
- [stl_function.h](#)

5.574 `std::num_get< _CharT, _InIter >` Class Template Reference

Primary class template [num_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.

Inheritance diagram for `std::num_get< _CharT, _InIter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_InIter` [iter_type](#)

Public Member Functions

- [num_get](#) (`size_t __refs=0`)
- `template<typename _ValueT >`
`_InIter M_extract_int (_InIter __beg, _InIter __end, ios_base &__io, ios_base::iostate &__err, _ValueT &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, void *&__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, bool &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const`

- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, float &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, double &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__v) const`

Static Public Attributes

- static `locale::id id`

Protected Member Functions

- virtual `~num_get ()`
- `iter_type _M_extract_float (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const`
- `template<typename _ValueT > iter_type _M_extract_int (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const`
- `template<typename _CharT2 > __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, int >::__type _M_find (const _CharT2 *, size_t __len, _CharT2 __c) const`
- `template<typename _CharT2 > __gnu_cxx::__enable_if<!__is_char< _CharT2 >::__value, int >::__type _M_find (const _CharT2 *__zero, size_t __len, _CharT2 __c) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, float &) const`

- virtual [iter_type do_get](#) ([iter_type](#), [iter_type](#), [ios_base &](#), [ios_base::iostate &__err](#), [double &](#)) const
- virtual [iter_type do_get](#) ([iter_type](#), [iter_type](#), [ios_base &](#), [ios_base::iostate &__err](#), [long double &](#)) const
- virtual [iter_type do_get](#) ([iter_type](#), [iter_type](#), [ios_base &](#), [ios_base::iostate &__err](#), [void *&](#)) const

Static Protected Member Functions

- static [__c_locale _S_clone_c_locale](#) ([__c_locale &__cloc](#)) throw ()
- static void [_S_create_c_locale](#) ([__c_locale &__cloc](#), [const char *__s](#), [__c_locale __old=0](#))
- static void [_S_destroy_c_locale](#) ([__c_locale &__cloc](#))
- static [__c_locale _S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) [const char * _S_get_c_name](#) () throw ()
- static [__c_locale _S_lc_ctype_c_locale](#) ([__c_locale __cloc](#), [const char *__s](#))

Friends

- class [locale::_Impl](#)

5.574.1 Detailed Description

template<typename _CharT, typename _InIter> class std::num_get< _CharT, _InIter >

Primary class template [num_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators. The [num_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [num_get](#) facet.

Definition at line 1911 of file [locale_facets.h](#).

5.574.2 Member Typedef Documentation

5.574.2.1 template<typename _CharT , typename _InIter > typedef _CharT std::num_get< _CharT, _InIter >::char_type

Public typedefs.

Definition at line 1917 of file [locale_facets.h](#).

5.574.2.2 `template<typename _CharT, typename _InIter > typedef _InIter
std::num_get< _CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1918 of file locale_facets.h.

5.574.3 Constructor & Destructor Documentation

5.574.3.1 `template<typename _CharT, typename _InIter > std::num_get<
_CharT, _InIter >::num_get (size_t __refs = 0) [inline,
explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 1932 of file locale_facets.h.

5.574.3.2 `template<typename _CharT, typename _InIter > virtual
std::num_get< _CharT, _InIter >::~num_get () [inline,
protected, virtual]`

Destructor.

Definition at line 2101 of file locale_facets.h.

5.574.4 Member Function Documentation

5.574.4.1 `template<typename _CharT, typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, float &
__v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 686 of file locale_facets.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`.

5.574.4.2 `template<typename _CharT, typename _InIter> _InIter
 std::num_get<_CharT, _InIter>::do_get (iter_type __beg,
 iter_type __end, ios_base & __io, ios_base::iostate & __err, bool &
 __v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 590 of file locale_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

Referenced by `std::num_get<_CharT, _InIter>::get()`.

```
5.574.4.3 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long &  
__v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2169 of file locale_facets.h.

```
5.574.4.4 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned short & __v ) const [inline, protected,  
virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2174 of file locale_facets.h.

```
5.574.4.5  template<typename _CharT , typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned int & __v ) const  [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2179 of file locale_facets.h.

```
5.574.4.6  template<typename _CharT , typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned long & __v ) const  [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2184 of file locale_facets.h.

```
5.574.4.7 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long  
long & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2190 of file locale_facets.h.

5.574.4.8 `template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned long long & __v) const [inline, protected,
virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2195 of file locale_facets.h.

5.574.4.9 `template<typename _CharT, typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long
double & __v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 733 of file locale_facets.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

```
5.574.4.10 template<typename _CharT , typename _InIter > _InIter  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
double & __v ) const [protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 701 of file locale_facets.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

5.574.4.11 `template<typename _CharT, typename _InIter> _InIter
std::num_get<_CharT, _InIter>::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, void
*& __v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 748 of file `locale_facets.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, and `std::ios_base::hex`.

5.574.4.12 `template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long & __v)
const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios_base::basefield](#). If equal to [ios_base::oct](#), parses like the `scanf` `o` specifier. Else if equal to [ios_base::hex](#), parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 1994 of file locale_facets.h.

References [std::num_get< _CharT, _InIter >::do_get\(\)](#).

5.574.4.13 `template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, double & __v)
const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.

err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2058 of file locale_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.574.4.14 template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, void *& __v )
const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf p` specifier.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2095 of file locale_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

5.574.4.15 `template<typename _CharT, typename _InIter> iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned long
& __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios_base::basefield](#). If equal to [ios_base::oct](#), parses like the `scanf` `o` specifier. Else if equal to [ios_base::hex](#), parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets `err` to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to [ios_base::failbit](#) and leaves *v* unaltered. Sets `err` to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2009 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

5.574.4.16 `template<typename _CharT, typename _InIter> iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, bool & __v)
const [inline]`

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 1958 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

Referenced by `std::basic_istream<_CharT, _Traits>::operator>>()`.

```
5.574.4.17  template<typename _CharT, typename _InIter> iter_type
             std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type
             __end, ios_base & __io, ios_base::iostate & __err, unsigned long
             long & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for v , v is set. Otherwise, sets err to [ios_base::failbit](#) and leaves v unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2020 of file locale_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.574.4.18 template<typename _CharT , typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, float & __v )  
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable v . It does so by calling [num_get::do_get\(\)](#).

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for v , v is set. Otherwise, sets err to [ios_base::failbit](#) and leaves v unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2053 of file locale_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

5.574.4.19 `template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned int &
__v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2004 of file locale_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

5.574.4.20 `template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long double &
__v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets *err* to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to [ios_base::failbit](#) and leaves *v* unaltered. Sets *err* to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2063 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

5.574.4.21 `template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long long & __v
) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

- Iterator after reading.

Definition at line 2015 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.574.4.22 template<typename _CharT , typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned short
& __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 1999 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

5.574.5 Member Data Documentation

5.574.5.1 `template<typename _CharT, typename _InIter> locale::id
std::num_get< _CharT, _InIter >::id [static]`

Numpunct facet id.

Definition at line 1922 of file `locale_facets.h`.

The documentation for this class was generated from the following files:

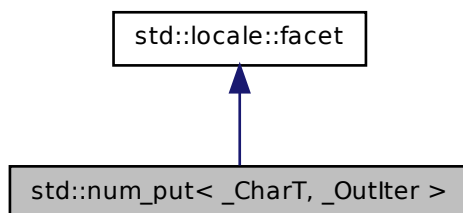
- [locale_facets.h](#)
- [locale_facets.tcc](#)

5.575 `std::num_put< _CharT, _OutIter >` Class Template Reference

Primary class template [num_put](#).

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

Inheritance diagram for `std::num_put< _CharT, _OutIter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_OutIter` [iter_type](#)

Public Member Functions

- [num_put](#) (size_t __refs=0)
- template<typename _ValueT >
`_OutIter` [_M_insert_float](#) (`_OutIter` __s, [ios_base](#) &__io, `_CharT` __fill, char __mod, _ValueT __v) const
- template<typename _ValueT >
`_OutIter` [_M_insert_int](#) (`_OutIter` __s, [ios_base](#) &__io, `_CharT` __fill, _ValueT __v) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__f, [char_type](#) __fill, const void *__v) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__f, [char_type](#) __fill, bool __v) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__f, [char_type](#) __fill, long __v) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__f, [char_type](#) __fill, unsigned long __v) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__f, [char_type](#) __fill, long long __v) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__f, [char_type](#) __fill, unsigned long long __v) const

- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, double __v) const`
- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, long double __v) const`

Static Public Attributes

- static `locale::id id`

Protected Member Functions

- virtual `~num_put ()`
- void `_M_group_float (const char *__grouping, size_t __grouping_size, char_type __sep, const char_type *__p, char_type *__new, char_type *__cs, int &__len) const`
- void `_M_group_int (const char *__grouping, size_t __grouping_size, char_type __sep, ios_base &__io, char_type *__new, char_type *__cs, int &__len) const`
- template<typename _ValueT >
`iter_type _M_insert_float (iter_type, ios_base &__io, char_type __fill, char __mod, _ValueT __v) const`
- template<typename _ValueT >
`iter_type _M_insert_int (iter_type, ios_base &__io, char_type __fill, _ValueT __v) const`
- void `_M_pad (char_type __fill, streamsize __w, ios_base &__io, char_type *__new, const char_type *__cs, int &__len) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, bool __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, double __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, long double __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, const void *__v) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.575.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::num_put< _CharT, _OutIter >`

Primary class template [num_put](#).

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The [num_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [num_put](#) facet.

Definition at line 2249 of file `locale_facets.h`.

5.575.2 Member Typedef Documentation

5.575.2.1 `template<typename _CharT , typename _OutIter > typedef _CharT std::num_put< _CharT, _OutIter >::char_type`

Public typedefs.

Definition at line 2255 of file `locale_facets.h`.

5.575.2.2 `template<typename _CharT , typename _OutIter > typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 2256 of file `locale_facets.h`.

5.575.3 Constructor & Destructor Documentation

5.575.3.1 `template<typename _CharT, typename _OutIter> std::num_put<_CharT, _OutIter>::num_put (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 2270 of file locale_facets.h.

5.575.3.2 `template<typename _CharT, typename _OutIter> virtual std::num_put<_CharT, _OutIter>::~~num_put () [inline, protected, virtual]`

Destructor.

Definition at line 2449 of file locale_facets.h.

5.575.4 Member Function Documentation

5.575.4.1 `template<typename _CharT, typename _OutIter> virtual iter_type std::num_put<_CharT, _OutIter>::do_put (iter_type __s, ios_base & __io, char_type __fill, unsigned long __v) const [inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2473 of file locale_facets.h.

5.575.4.2 `template<typename _CharT, typename _OutIter > virtual
iter_type std::num_put< _CharT, _OutIter >::do_put (iter_type
__s, ios_base & __io, char_type __fill, long long __v) const
[inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2479 of file locale_facets.h.

5.575.4.3 `template<typename _CharT, typename _OutIter > virtual iter_type
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, long __v) const [inline,
protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2469 of file locale_facets.h.


```

5.575.4.4 template<typename _CharT, typename _OutIter> virtual iter_type
std::num_put<_CharT, _OutIter>::do_put ( iter_type __s,
ios_base & __io, char_type __fill, unsigned long long __v ) const
[inline, protected, virtual]

```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2484 of file locale_facets.h.

```

5.575.4.5 template<typename _CharT, typename _OutIter> _OutIter
std::num_put<_CharT, _OutIter>::do_put ( iter_type __s,
ios_base & __io, char_type __fill, bool __v ) const [protected,
virtual]

```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1089 of file locale_facets.tcc.

References std::ios_base::_M_getloc(), std::ios_base::adjustfield, std::ios_base::boolalpha, std::ios_base::flags(), std::ios_base::left, and std::ios_base::width().

Referenced by std::num_put<_CharT, _OutIter>::put().

5.575.4.6 `template<typename _CharT, typename _OutIter > _OutIter
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, long double __v) const
[protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1155 of file locale_facets.tcc.

5.575.4.7 `template<typename _CharT, typename _OutIter > _OutIter
std::num_put< _CharT, _OutIter >::do_put (iter_type
__s, ios_base & __io, char_type __fill, double __v) const
[protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1141 of file locale_facets.tcc.

```

5.575.4.8 template<typename _CharT, typename _OutIter> _OutIter
std::num_put<_CharT, _OutIter>::do_put ( iter_type __s,
ios_base & __io, char_type __fill, const void * __v ) const
[protected, virtual]

```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1162 of file locale_facets.tcc.

References `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::uppercase`.

```

5.575.4.9 template<typename _CharT, typename _OutIter> iter_type
std::num_put<_CharT, _OutIter>::put ( iter_type __s, ios_base &
__f, char_type __fill, long __v ) const [inline]

```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf` `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If [ios_base::internal](#), then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2330 of file `locale_facets.h`.

References `std::num_put<_CharT, _OutIter >::do_put()`.

5.575.4.10 `template<typename _CharT, typename _OutIter> iter_type
std::num_put<_CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, long long __v) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to [ios_base::oct](#), formats like the printf 'o' specifier. Else if equal to [ios_base::hex](#), formats like x or X with [ios_base::uppercase](#) unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If [ios_base::showpos](#) is set, '+' is output before positive values. If [ios_base::showbase](#) is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If [ios_base::internal](#), then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2340 of file locale_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

5.575.4.11 `template<typename _CharT , typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, unsigned long __v) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf o specifier. Else if equal to `ios_base::hex`, formats like x or X with `ios_base::uppercase` unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[X]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2334 of file locale_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

5.575.4.12 `template<typename _CharT, typename _OutIter > iter_type
std::num_put<_CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, bool __v) const [inline]`

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truenam()` or `ctype<CharT>::falsenam()`. Otherwise formats *v* as an int.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2288 of file locale_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

5.575.4.13 `template<typename _CharT, typename _OutIter > iter_type
std::num_put<_CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, long double __v) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2397 of file `locale_facets.h`.

References `std::num_put< _CharT, _OutIter >::do_put()`.

5.575.4.14 `template<typename _CharT, typename _OutIter> iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, const void * __v) const [inline]`

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats *v* as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

Parameters

s Stream to write to.
io Source of locale and flags.
fill Char_type to use for filling.
v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2418 of file locale_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

5.575.4.15 `template<typename _CharT, typename _OutIter> iter_type
 std::num_put<_CharT, _OutIter>::put (iter_type __s, ios_base
 & __f, char_type __fill, unsigned long long __v) const
 [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf` `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

s Stream to write to.
io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2344 of file locale_facets.h.

References std::num_put< _CharT, _OutIter >::do_put().

5.575.4.16 `template<typename _CharT, typename _OutIter> iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, double __v) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling [num_put::do_put\(\)](#).

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of *io.flags()* & [ios_base::floatfield](#). If equal to [ios_base::fixed](#), formats like the printf *f* specifier. Else if equal to [ios_base::scientific](#), formats like *e* or *E* with [ios_base::uppercase](#) unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by *io.precision()*. This precision is capped at [numeric_limits::digits10](#) + 2 (different for double and long double). The default precision is 6.

If [ios_base::showpos](#) is set, '+' is output before positive values. If [ios_base::showpoint](#) is set, a decimal point will always be output.

Thousands separators are inserted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). The decimal point character used is [numpunct::decimal_point\(\)](#).

If *io.width()* is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (*io.flags()* & [ios_base::adjustfield](#)) == [ios_base::left](#), result is padded at the end. If [ios_base::internal](#), then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2393 of file locale_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

5.575.5 Member Data Documentation

5.575.5.1 `template<typename _CharT, typename _OutIter > locale::id
std::num_put<_CharT, _OutIter >::id [static]`

Numpunct facet id.

Definition at line 2260 of file locale_facets.h.

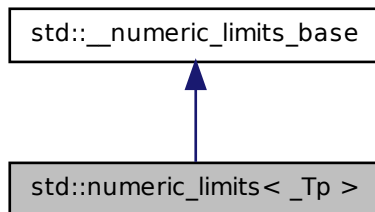
The documentation for this class was generated from the following files:

- [locale_facets.h](#)
- [locale_facets.tcc](#)

5.576 std::numeric_limits< _Tp > Struct Template Reference

Properties of fundamental types.

Inheritance diagram for `std::numeric_limits< _Tp >`:



Static Public Member Functions

- static _Tp [denorm_min](#) () throw ()
- static _Tp [epsilon](#) () throw ()
- static _Tp [infinity](#) () throw ()
- static _Tp [lowest](#) () throw ()
- static _Tp [max](#) () throw ()
- static _Tp [min](#) () throw ()
- static _Tp [quiet_NaN](#) () throw ()
- static _Tp [round_error](#) () throw ()
- static _Tp [signaling_NaN](#) () throw ()

Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const [float_denorm_style](#) [has_denorm](#)
- static const bool [has_denorm_loss](#)
- static const bool [has_infinity](#)
- static const bool [has_quiet_NaN](#)
- static const bool [has_signaling_NaN](#)
- static const bool [is_bounded](#)
- static const bool [is_exact](#)
- static const bool [is_iec559](#)
- static const bool [is_integer](#)
- static const bool [is_modulo](#)
- static const bool [is_signed](#)
- static const bool [is_specialized](#)
- static const int [max_digits10](#)
- static const int [max_exponent](#)
- static const int [max_exponent10](#)
- static const int [min_exponent](#)
- static const int [min_exponent10](#)
- static const int [radix](#)
- static const [float_round_style](#) [round_style](#)
- static const bool [tinyness_before](#)
- static const bool [traps](#)

5.576.1 Detailed Description

template<typename _Tp> struct std::numeric_limits< _Tp >

Properties of fundamental types. This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS`: DRs 201 and 184 (hi Gaby!) are noted, but not incorporated in this documented (yet).

Definition at line 284 of file `limits`.

5.576.2 Member Function Documentation

5.576.2.1 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::denorm_min () throw () [inline, static]**

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

Definition at line 313 of file `limits`.

5.576.2.2 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::epsilon () throw () [inline, static]**

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 298 of file `limits`.

5.576.2.3 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::infinity () throw () [inline, static]**

The representation of positive infinity, if `has_infinity`.

Definition at line 302 of file `limits`.

5.576.2.4 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::lowest () throw () [inline, static]**

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 294 of file `limits`.

5.576.2.5 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::max () throw () [inline, static]`

The maximum finite value.

Definition at line 290 of file limits.

5.576.2.6 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::min () throw () [inline, static]`

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 288 of file limits.

5.576.2.7 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::quiet_NaN () throw () [inline, static]`

The representation of a quiet *Not a Number*, if has_quiet_NaN.

Definition at line 306 of file limits.

5.576.2.8 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::round_error () throw () [inline, static]`

The maximum rounding error measurement (see LIA-1).

Definition at line 300 of file limits.

5.576.2.9 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::signaling_NaN () throw () [inline, static]`

The representation of a signaling *Not a Number*, if has_signaling_NaN.

Definition at line 309 of file limits.

5.576.3 Member Data Documentation

5.576.3.1 `const int std::__numeric_limits_base::digits [static, inherited]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 199 of file limits.

5.576.3.2 `const int std::__numeric_limits_base::digits10 [static, inherited]`

The number of base 10 digits that can be represented without change.

Definition at line 201 of file limits.

5.576.3.3 `const float_denorm_style std::__numeric_limits_base::has_denorm [static, inherited]`

See [std::float_denorm_style](#) for more information.

Definition at line 244 of file limits.

5.576.3.4 `const bool std::__numeric_limits_base::has_denorm_loss [static, inherited]`

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42

Definition at line 247 of file limits.

5.576.3.5 `const bool std::__numeric_limits_base::has_infinity [static, inherited]`

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

5.576.3.6 `const bool std::__numeric_limits_base::has_quiet_NaN [static, inherited]`

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 239 of file limits.

5.576.3.7 `const bool std::__numeric_limits_base::has_signaling_NaN [static, inherited]`

True if the type has a representation for a signaling *Not a Number*.

Definition at line 242 of file limits.

5.576.3.8 const bool std::__numeric_limits_base::is_bounded [static, inherited]

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54

Definition at line 255 of file limits.

5.576.3.9 const bool std::__numeric_limits_base::is_exact [static, inherited]

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15

Definition at line 216 of file limits.

5.576.3.10 const bool std::__numeric_limits_base::is_iec559 [static, inherited]

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 251 of file limits.

5.576.3.11 const bool std::__numeric_limits_base::is_integer [static, inherited]

True if the type is integer. Is this supposed to be if the type is integral?

Definition at line 211 of file limits.

5.576.3.12 const bool std::__numeric_limits_base::is_modulo [static, inherited]

True if the type is modulo, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 260 of file limits.

5.576.3.13 const bool std::__numeric_limits_base::is_signed [static, inherited]

True if the type is signed.

Definition at line 208 of file limits.

5.576.3.14 `const bool std::__numeric_limits_base::is_specialized [static, inherited]`

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 194 of file limits.

5.576.3.15 `const int std::__numeric_limits_base::max_digits10 [static, inherited]`

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 205 of file limits.

5.576.3.16 `const int std::__numeric_limits_base::max_exponent [static, inherited]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 230 of file limits.

5.576.3.17 `const int std::__numeric_limits_base::max_exponent10 [static, inherited]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 233 of file limits.

5.576.3.18 `const int std::__numeric_limits_base::min_exponent [static, inherited]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 223 of file limits.

5.576.3.19 `const int std::__numeric_limits_base::min_exponent10` `[static, inherited]`

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 226 of file limits.

5.576.3.20 `const int std::__numeric_limits_base::radix` `[static, inherited]`

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 219 of file limits.

5.576.3.21 `const float_round_style std::__numeric_limits_base::round_style` `[static, inherited]`

See [std::float_round_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 269 of file limits.

5.576.3.22 `const bool std::__numeric_limits_base::tinyness_before` `[static, inherited]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file limits.

5.576.3.23 `const bool std::__numeric_limits_base::traps` `[static, inherited]`

True if trapping is implemented for this type.

Definition at line 263 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.577 `std::numeric_limits< bool >` Struct Template Reference

[numeric_limits<bool>](#) specialization.

Static Public Member Functions

- static bool **denorm_min** () throw ()
- static bool **epsilon** () throw ()
- static bool **infinity** () throw ()
- static bool **lowest** () throw ()
- static bool **max** () throw ()
- static bool **min** () throw ()
- static bool **quiet_NaN** () throw ()
- static bool **round_error** () throw ()
- static bool **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.577.1 Detailed Description

`template<> struct std::numeric_limits< bool >`

[numeric_limits<bool>](#) specialization.

Definition at line 335 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.578 `std::numeric_limits< char >` Struct Template Reference

[numeric_limits<char>](#) specialization.

Static Public Member Functions

- static char **denorm_min** () throw ()
- static char **epsilon** () throw ()
- static char **infinity** () throw ()
- static char **lowest** () throw ()
- static char **max** () throw ()
- static char **min** () throw ()
- static char **quiet_NaN** () throw ()
- static char **round_error** () throw ()
- static char **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.578.1 Detailed Description

`template<> struct std::numeric_limits< char >`

[numeric_limits<char>](#) specialization.

Definition at line 395 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.579 `std::numeric_limits< char16_t >` Struct Template Reference

[numeric_limits<char16_t>](#) specialization.

Static Public Member Functions

- static `char16_t` **denorm_min** () throw ()
- static `char16_t` **epsilon** () throw ()
- static `char16_t` **infinity** () throw ()
- static `char16_t` **lowest** () throw ()
- static `char16_t` **max** () throw ()
- static `char16_t` **min** () throw ()
- static `char16_t` **quiet_NaN** () throw ()
- static `char16_t` **round_error** () throw ()
- static `char16_t` **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.579.1 Detailed Description

`template<> struct std::numeric_limits< char16_t >`

[numeric_limits<char16_t>](#) specialization.

Definition at line 628 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.580 `std::numeric_limits< char32_t >` Struct Template Reference

[numeric_limits<char32_t>](#) specialization.

Static Public Member Functions

- static char32_t **denorm_min** () throw ()
- static char32_t **epsilon** () throw ()
- static char32_t **infinity** () throw ()
- static char32_t **lowest** () throw ()
- static char32_t **max** () throw ()
- static char32_t **min** () throw ()
- static char32_t **quiet_NaN** () throw ()
- static char32_t **round_error** () throw ()
- static char32_t **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.580.1 Detailed Description

`template<> struct std::numeric_limits< char32_t >`

[numeric_limits<char32_t>](#) specialization.

Definition at line 686 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.581 `std::numeric_limits< double >` Struct Template Reference

[numeric_limits<double>](#) specialization.

Static Public Member Functions

- static double **denorm_min** () throw ()
- static double **epsilon** () throw ()
- static double **infinity** () throw ()
- static double **lowest** () throw ()
- static double **max** () throw ()
- static double **min** () throw ()
- static double **quiet_NaN** () throw ()
- static double **round_error** () throw ()
- static double **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.581.1 Detailed Description

`template<> struct std::numeric_limits< double >`

[numeric_limits<double>](#) specialization.

Definition at line 1274 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.582 `std::numeric_limits< float >` Struct Template Reference

[numeric_limits<float>](#) specialization.

Static Public Member Functions

- static float **denorm_min** () throw ()
- static float **epsilon** () throw ()
- static float **infinity** () throw ()
- static float **lowest** () throw ()
- static float **max** () throw ()
- static float **min** () throw ()
- static float **quiet_NaN** () throw ()
- static float **round_error** () throw ()
- static float **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.582.1 Detailed Description

`template<> struct std::numeric_limits< float >`

[numeric_limits<float>](#) specialization.

Definition at line 1209 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.583 `std::numeric_limits< int >` Struct Template Reference

[numeric_limits<int>](#) specialization.

Static Public Member Functions

- static int **denorm_min** () throw ()
- static int **epsilon** () throw ()
- static int **infinity** () throw ()
- static int **lowest** () throw ()
- static int **max** () throw ()
- static int **min** () throw ()
- static int **quiet_NaN** () throw ()
- static int **round_error** () throw ()
- static int **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.583.1 Detailed Description

`template<> struct std::numeric_limits< int >`

[numeric_limits<int>](#) specialization.

Definition at line 861 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.584 `std::numeric_limits< long >` Struct Template Reference

[numeric_limits<long>](#) specialization.

Static Public Member Functions

- static long **denorm_min** () throw ()
- static long **epsilon** () throw ()
- static long **infinity** () throw ()
- static long **lowest** () throw ()
- static long **max** () throw ()
- static long **min** () throw ()
- static long **quiet_NaN** () throw ()
- static long **round_error** () throw ()
- static long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.584.1 Detailed Description

`template<> struct std::numeric_limits< long >`

[numeric_limits<long>](#) specialization.

Definition at line 977 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.585 `std::numeric_limits< long double >` Struct Template Reference

[numeric_limits<long double>](#) specialization.

Static Public Member Functions

- static long double **denorm_min** () throw ()
- static long double **epsilon** () throw ()
- static long double **infinity** () throw ()
- static long double **lowest** () throw ()
- static long double **max** () throw ()
- static long double **min** () throw ()
- static long double **quiet_NaN** () throw ()
- static long double **round_error** () throw ()
- static long double **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.585.1 Detailed Description

`template<> struct std::numeric_limits< long double >`

[numeric_limits<long double>](#) specialization.

Definition at line 1339 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.586 `std::numeric_limits< long long >` Struct Template Reference

[numeric_limits<long long>](#) specialization.

Static Public Member Functions

- static long long **denorm_min** () throw ()
- static long long **epsilon** () throw ()
- static long long **infinity** () throw ()
- static long long **lowest** () throw ()
- static long long **max** () throw ()
- static long long **min** () throw ()
- static long long **quiet_NaN** () throw ()
- static long long **round_error** () throw ()
- static long long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.586.1 Detailed Description

`template<> struct std::numeric_limits< long long >`

[numeric_limits<long long>](#) specialization.

Definition at line 1093 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.587 `std::numeric_limits< short >` Struct Template Reference

[numeric_limits<short>](#) specialization.

Static Public Member Functions

- static short **denorm_min** () throw ()
- static short **epsilon** () throw ()
- static short **infinity** () throw ()
- static short **lowest** () throw ()
- static short **max** () throw ()
- static short **min** () throw ()
- static short **quiet_NaN** () throw ()
- static short **round_error** () throw ()
- static short **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.587.1 Detailed Description

`template<> struct std::numeric_limits< short >`

[numeric_limits<short>](#) specialization.

Definition at line 745 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.588 `std::numeric_limits< signed char >` Struct Template Reference

[numeric_limits<signed char>](#) specialization.

Static Public Member Functions

- static signed char **denorm_min** () throw ()
- static signed char **epsilon** () throw ()
- static signed char **infinity** () throw ()
- static signed char **lowest** () throw ()
- static signed char **max** () throw ()
- static signed char **min** () throw ()
- static signed char **quiet_NaN** () throw ()
- static signed char **round_error** () throw ()
- static signed char **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.588.1 Detailed Description

`template<> struct std::numeric_limits< signed char >`

[numeric_limits<signed char>](#) specialization.

Definition at line 453 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.589 `std::numeric_limits< unsigned char >` Struct Template Reference

[numeric_limits<unsigned char>](#) specialization.

Static Public Member Functions

- static unsigned char **denorm_min** () throw ()
- static unsigned char **epsilon** () throw ()
- static unsigned char **infinity** () throw ()
- static unsigned char **lowest** () throw ()
- static unsigned char **max** () throw ()
- static unsigned char **min** () throw ()
- static unsigned char **quiet_NaN** () throw ()
- static unsigned char **round_error** () throw ()
- static unsigned char **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.589.1 Detailed Description

`template<> struct std::numeric_limits< unsigned char >`

[numeric_limits<unsigned char>](#) specialization.

Definition at line 511 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.590 `std::numeric_limits< unsigned int >` Struct Template Reference

[numeric_limits<unsigned int>](#) specialization.

Static Public Member Functions

- static unsigned int **denorm_min** () throw ()
- static unsigned int **epsilon** () throw ()
- static unsigned int **infinity** () throw ()
- static unsigned int **lowest** () throw ()
- static unsigned int **max** () throw ()
- static unsigned int **min** () throw ()
- static unsigned int **quiet_NaN** () throw ()
- static unsigned int **round_error** () throw ()
- static unsigned int **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.590.1 Detailed Description

`template<> struct std::numeric_limits< unsigned int >`

[numeric_limits<unsigned int>](#) specialization.

Definition at line 919 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.591 `std::numeric_limits< unsigned long >` Struct Template Reference

[numeric_limits<unsigned long>](#) specialization.

Static Public Member Functions

- static unsigned long **denorm_min** () throw ()
- static unsigned long **epsilon** () throw ()
- static unsigned long **infinity** () throw ()
- static unsigned long **lowest** () throw ()
- static unsigned long **max** () throw ()
- static unsigned long **min** () throw ()
- static unsigned long **quiet_NaN** () throw ()
- static unsigned long **round_error** () throw ()
- static unsigned long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.591.1 Detailed Description

template<> struct std::numeric_limits< unsigned long >

[numeric_limits<unsigned long>](#) specialization.

Definition at line 1035 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.592 std::numeric_limits< unsigned long long > Struct Template Reference

[numeric_limits<unsigned long long>](#) specialization.

Static Public Member Functions

- static unsigned long long **denorm_min** () throw ()
- static unsigned long long **epsilon** () throw ()
- static unsigned long long **infinity** () throw ()
- static unsigned long long **lowest** () throw ()
- static unsigned long long **max** () throw ()
- static unsigned long long **min** () throw ()
- static unsigned long long **quiet_NaN** () throw ()
- static unsigned long long **round_error** () throw ()
- static unsigned long long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.592.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long long >`

[numeric_limits<unsigned long long>](#) specialization.

Definition at line 1151 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.593 `std::numeric_limits< unsigned short >` Struct Template Reference

[numeric_limits<unsigned short>](#) specialization.

Static Public Member Functions

- static unsigned short **denorm_min** () throw ()
- static unsigned short **epsilon** () throw ()
- static unsigned short **infinity** () throw ()
- static unsigned short **lowest** () throw ()
- static unsigned short **max** () throw ()
- static unsigned short **min** () throw ()
- static unsigned short **quiet_NaN** () throw ()
- static unsigned short **round_error** () throw ()
- static unsigned short **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.593.1 Detailed Description

`template<> struct std::numeric_limits< unsigned short >`

[numeric_limits<unsigned short>](#) specialization.

Definition at line 803 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.594 `std::numeric_limits< wchar_t >` Struct Template Reference

[numeric_limits<wchar_t>](#) specialization.

Static Public Member Functions

- static `wchar_t` **denorm_min** () throw ()
- static `wchar_t` **epsilon** () throw ()
- static `wchar_t` **infinity** () throw ()
- static `wchar_t` **lowest** () throw ()
- static `wchar_t` **max** () throw ()
- static `wchar_t` **min** () throw ()
- static `wchar_t` **quiet_NaN** () throw ()
- static `wchar_t` **round_error** () throw ()
- static `wchar_t` **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.594.1 Detailed Description

template<> struct std::numeric_limits< wchar_t >

[numeric_limits<wchar_t>](#) specialization.

Definition at line 569 of file limits.

The documentation for this struct was generated from the following file:

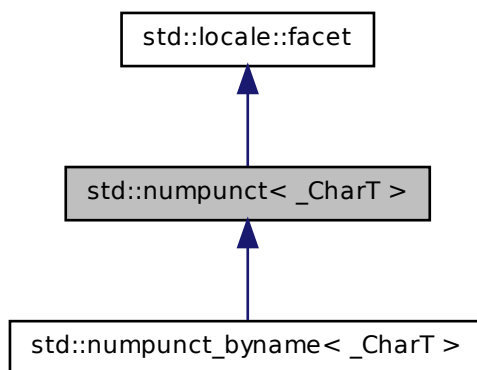
- [limits](#)

5.595 std::num_punct<_CharT> Class Template Reference

Primary class template num_punct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

Inheritance diagram for `std::numpunct<_CharT>`:



Public Types

- `typedef __numpunct_cache<_CharT> __cache_type`
- `typedef _CharT char_type`
- `typedef basic_string<_CharT> string_type`

Public Member Functions

- `numpunct` (`size_t __refs=0`)
- `numpunct` (`__cache_type *__cache, size_t __refs=0`)
- `numpunct` (`__c_locale __cloc, size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~num_punct](#) ()
- template<>
void [_M_initialize_num_punct](#) (__c_locale __cloc)
- template<>
void [_M_initialize_num_punct](#) (__c_locale __cloc)
- void [_M_initialize_num_punct](#) (__c_locale __cloc=0)
- virtual [char_type do_decimal_point](#) () const
- virtual [string_type do_falsename](#) () const
- virtual [string do_grouping](#) () const
- virtual [char_type do_thousands_sep](#) () const
- virtual [string_type do_truename](#) () const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * [_M_data](#)

Friends

- class [locale::_Impl](#)

5.595.1 Detailed Description

template<typename _CharT> class std::numpunct< _CharT >

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers. The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

Definition at line 1637 of file locale_facets.h.

5.595.2 Member Typedef Documentation

5.595.2.1 template<typename _CharT > typedef _CharT std::numpunct< _CharT >::char_type

Public typedefs.

Reimplemented in [std::numpunct_byname< _CharT >](#).

Definition at line 1643 of file locale_facets.h.

5.595.2.2 template<typename _CharT > typedef basic_string<_CharT> std::numpunct< _CharT >::string_type

Public typedefs.

Reimplemented in [std::numpunct_byname< _CharT >](#).

Definition at line 1644 of file locale_facets.h.

5.595.3 Constructor & Destructor Documentation

5.595.3.1 template<typename _CharT > std::numpunct< _CharT >::numpunct (size_t __refs = 0) [inline, explicit]

Numpunct constructor.

Parameters

refs Refcount to pass to the base class.

Definition at line 1661 of file locale_facets.h.

5.595.3.2 `template<typename _CharT> std::num_punct<_CharT>
>::num_punct (__cache_type * __cache, size_t __refs = 0)
[inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

Parameters

cache __num_punct_cache object.

refs Refcount to pass to the base class.

Definition at line 1675 of file locale_facets.h.

5.595.3.3 `template<typename _CharT> std::num_punct<_CharT>
>::num_punct (__c_locale __cloc, size_t __refs = 0) [inline,
explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.

refs Refcount to pass to the base class.

Definition at line 1689 of file locale_facets.h.

5.595.3.4 `template<typename _CharT> virtual std::num_punct<_CharT>
>::~~num_punct () [protected, virtual]`

Destructor.

5.595.4 Member Function Documentation

5.595.4.1 `template<typename _CharT> char_type std::num_punct<_CharT>
>::decimal_point () const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `numpunct<char_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1703 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_decimal_point()`.

5.595.4.2 `template<typename _CharT> virtual char_type std::numpunct<_CharT>::do_decimal_point() const [inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1790 of file `locale_facets.h`.

Referenced by `std::numpunct<_CharT>::decimal_point()`.

5.595.4.3 `template<typename _CharT> virtual string_type std::numpunct<_CharT>::do_falsename() const [inline, protected, virtual]`

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of false.

Definition at line 1841 of file `locale_facets.h`.

Referenced by `std::numpunct<_CharT>::falsename()`.

5.595.4.4 `template<typename _CharT > virtual string std::num_punct< _CharT >::do_grouping() const [inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1815 of file locale_facets.h.

Referenced by `std::num_punct< _CharT >::grouping()`.

5.595.4.5 `template<typename _CharT > virtual char_type std::num_punct< _CharT >::do_thousands_sep() const [inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1802 of file locale_facets.h.

Referenced by `std::num_punct< _CharT >::thousands_sep()`.

5.595.4.6 `template<typename _CharT > virtual string_type std::num_punct< _CharT >::do_truename() const [inline, protected, virtual]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of true.

Definition at line 1828 of file locale_facets.h.

Referenced by `std::num_punct<_CharT>::true_name()`.

5.595.4.7 `template<typename _CharT> string_type std::num_punct<_CharT>::false_name () const [inline]`

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_false_name()`.

Returns

`string_type` representing printed form of false.

Definition at line 1773 of file locale_facets.h.

References `std::num_punct<_CharT>::do_false_name()`.

5.595.4.8 `template<typename _CharT> string std::num_punct<_CharT>::grouping () const [inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1747 of file locale_facets.h.

References `std::num_punct<_CharT>::do_grouping()`.

5.595.4.9 `template<typename _CharT> char_type std::num_punct<_CharT>::thousands_sep() const [inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning returning `num_punct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1716 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_thousands_sep()`.

5.595.4.10 `template<typename _CharT> string_type std::num_punct<_CharT>::true_name() const [inline]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `num_punct<char_type>::do_true_name()`.

Returns

`string_type` representing printed form of true.

Definition at line 1760 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_true_name()`.

5.595.5 Member Data Documentation

5.595.5.1 `template<typename _CharT> locale::id std::num_punct<_CharT>::id [static]`

Numpunct facet id.

Definition at line 1653 of file `locale_facets.h`.

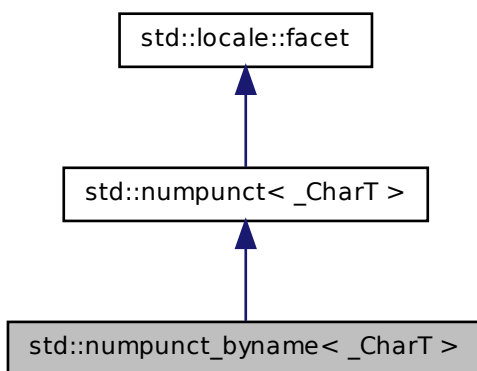
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.596 std::numpunct_byname<_CharT> Class Template Reference

class [numpunct_byname](#) [22.2.3.2].

Inheritance diagram for std::numpunct_byname<_CharT>:



Public Types

- typedef __numpunct_cache<_CharT> __cache_type
- typedef _CharT [char_type](#)
- typedef [basic_string](#)<_CharT> [string_type](#)

Public Member Functions

- **numpunct_byname** (const char *__s, size_t __refs=0)
- [char_type decimal_point](#) () const
- [string_type falsename](#) () const
- [string grouping](#) () const
- [char_type thousands_sep](#) () const
- [string_type truename](#) () const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- `template<>`
void [_M_initialize_numpunct](#) (__c_locale __cloc)
- `template<>`
void [_M_initialize_numpunct](#) (__c_locale __cloc)
- void [_M_initialize_numpunct](#) (__c_locale __cloc=0)
- virtual [char_type do_decimal_point](#) () const
- virtual [string_type do_falsename](#) () const
- virtual [string do_grouping](#) () const
- virtual [char_type do_thousands_sep](#) () const
- virtual [string_type do_truename](#) () const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- `__cache_type * _M_data`

Friends

- class [locale::_Impl](#)

5.596.1 Detailed Description

`template<typename _CharT> class std::numpunct_byname<_CharT>`

class [numpunct_byname](#) [22.2.3.2].

Definition at line 1870 of file `locale_facets.h`.

5.596.2 Member Typedef Documentation

5.596.2.1 `template<typename _CharT > typedef _CharT std::num_punct_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::num_punct< _CharT >](#).

Definition at line 1873 of file locale_facets.h.

5.596.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::num_punct_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::num_punct< _CharT >](#).

Definition at line 1874 of file locale_facets.h.

5.596.3 Member Function Documentation

5.596.3.1 `template<typename _CharT > char_type std::num_punct< _CharT >::decimal_point() const [inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning [num_punct<char_type>::do_decimal_point\(\)](#).

Returns

char_type representing a decimal point.

Definition at line 1703 of file locale_facets.h.

References [std::num_punct< _CharT >::do_decimal_point\(\)](#).

5.596.3.2 `template<typename _CharT > virtual char_type std::num_punct< _CharT >::do_decimal_point() const [inline, protected, virtual, inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1790 of file locale_facets.h.

Referenced by std::num_punct< _CharT >::decimal_point().

5.596.3.3 template<typename _CharT > virtual string_type std::num_punct< _CharT >::do_falsename () const [inline, protected, virtual, inherited]

Return string representation of bool false.

Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Definition at line 1841 of file locale_facets.h.

Referenced by std::num_punct< _CharT >::falsename().

5.596.3.4 template<typename _CharT > virtual string std::num_punct< _CharT >::do_grouping () const [inline, protected, virtual, inherited]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1815 of file locale_facets.h.

Referenced by std::num_punct< _CharT >::grouping().

5.596.3.5 `template<typename _CharT > virtual char_type std::num_punct<_CharT >::do_thousands_sep() const [inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1802 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT >::thousands_sep()`.

5.596.3.6 `template<typename _CharT > virtual string_type std::num_punct<_CharT >::do_truename() const [inline, protected, virtual, inherited]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Definition at line 1828 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT >::truename()`.

5.596.3.7 `template<typename _CharT > string_type std::num_punct<_CharT >::falsename() const [inline, inherited]`

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

Returns

string_type representing printed form of false.

Definition at line 1773 of file `locale_facets.h`.

References `std::num_punct<_CharT >::do_falsename()`.

5.596.3.8 `template<typename _CharT > string std::num_punct< _CharT >::grouping() const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1747 of file `locale_facets.h`.

References `std::num_punct< _CharT >::do_grouping()`.

5.596.3.9 `template<typename _CharT > char_type std::num_punct< _CharT >::thousands_sep() const [inline, inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `num_punct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1716 of file `locale_facets.h`.

References `std::num_punct< _CharT >::do_thousands_sep()`.

5.596.3.10 `template<typename _CharT > string_type std::num_punct< _CharT >::true_name() const [inline, inherited]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

Returns

`string_type` representing printed form of true.

Definition at line 1760 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_truename()`.

5.596.4 Member Data Documentation

5.596.4.1 `template<typename _CharT> locale::id std::numpunct<_CharT>::id [static, inherited]`

Numpunct facet id.

Definition at line 1653 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.597 `std::once_flag` Struct Reference

[once_flag](#)

Public Member Functions

- `once_flag` (const [once_flag](#) &)
- `once_flag` & `operator=` (const [once_flag](#) &)

Friends

- `template<typename _Callable, typename... _Args>`
void `call_once` ([once_flag](#) &__once, _Callable __f, _Args &&...__args)

5.597.1 Detailed Description

[once_flag](#)

Definition at line 675 of file `mutex`.

5.598 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference 2885

5.597.2 Friends And Related Function Documentation

5.597.2.1 `template<typename _Callable, typename... _Args> void call_once
(once_flag & __once, _Callable __f, _Args &&... __args)
[friend]`

call_once

Definition at line 721 of file mutex.

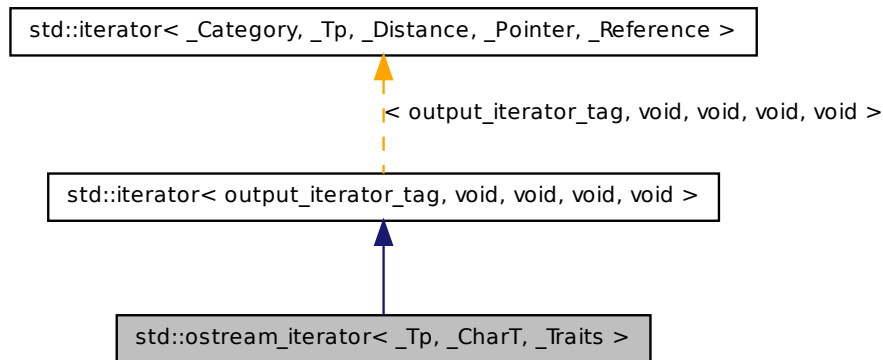
The documentation for this struct was generated from the following file:

- [mutex](#)

5.598 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference

Provides output iterator semantics for streams.

Inheritance diagram for std::ostream_iterator< _Tp, _CharT, _Traits >:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) `iterator_category`

- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [basic_ostream](#)< [_CharT](#), [_Traits](#) > [ostream_type](#)

Public Member Functions

- [ostream_iterator](#) ([ostream_type](#) &__s)
- [ostream_iterator](#) ([ostream_type](#) &__s, const [_CharT](#) *__c)
- [ostream_iterator](#) (const [ostream_iterator](#) &__obj)
- [ostream_iterator](#) & **operator*** ()
- [ostream_iterator](#) & **operator++** (int)
- [ostream_iterator](#) & **operator++** ()
- [ostream_iterator](#) & **operator=** (const [_Tp](#) &__value)

5.598.1 Detailed Description

template<typename [_Tp](#), typename [_CharT](#) = char, typename [_Traits](#) = [char_traits](#)<[_CharT](#)>> class std::ostream_iterator< [_Tp](#), [_CharT](#), [_Traits](#) >

Provides output iterator semantics for streams. This class provides an iterator to write to an ostream. The type [Tp](#) is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

Parameters

Tp The type to write to the ostream.

CharT The ostream [char_type](#).

Traits The ostream [char_traits](#).

Definition at line 152 of file [stream_iterator.h](#).

5.598.2 Member Typedef Documentation

5.598.2.1 template<typename [_Tp](#) , typename [_CharT](#) = char, typename [_Traits](#) = [char_traits](#)<[_CharT](#)>> typedef [_CharT](#) std::ostream_iterator< [_Tp](#), [_CharT](#), [_Traits](#) >::char_type

Public typedef.

Definition at line 158 of file [stream_iterator.h](#).

5.598 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference

5.598.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

5.598.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

5.598.2.4 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 160 of file stream_iterator.h.

5.598.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 116 of file stl_iterator_base_types.h.

5.598.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.598.2.7 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 159 of file stream_iterator.h.

5.598.2.8 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file `stl_iterator_base_types.h`.

5.598.3 Constructor & Destructor Documentation

5.598.3.1 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (ostream_type & __s) [inline]`

Construct from an ostream.

Definition at line 169 of file `stream_iterator.h`.

5.598.3.2 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (ostream_type & __s, const _CharT * __c) [inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

Parameters

s Underlying ostream to write to.

c *CharT* delimiter string to insert.

Definition at line 181 of file `stream_iterator.h`.

5.598.3.3 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (const ostream_iterator< _Tp, _CharT, _Traits > & __obj) [inline]`

Copy constructor.

Definition at line 185 of file `stream_iterator.h`.

5.598.4 Member Function Documentation

5.598.4.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> ostream_iterator& std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (const _Tp & __value) [inline]`

Writes *value* to underlying ostream using operator<<. If /// constructed with delimiter string, writes delimiter to ostream.

Definition at line 191 of file stream_iterator.h.

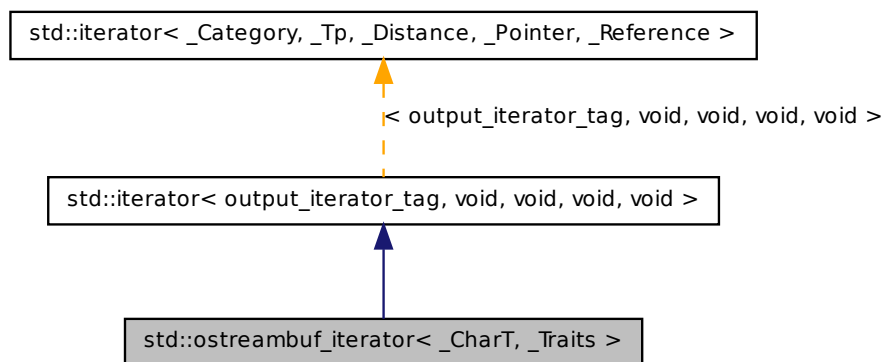
The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

5.599 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference

Provides output iterator semantics for streambufs.

Inheritance diagram for std::ostreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef void [difference_type](#)
 - typedef [output_iterator_tag](#) iterator_category
 - typedef void [pointer](#)
 - typedef void [reference](#)
 - typedef void [value_type](#)
-
- typedef [_CharT](#) [char_type](#)
 - typedef [_Traits](#) [traits_type](#)
 - typedef [basic_streambuf](#)< [_CharT](#), [_Traits](#) > [streambuf_type](#)
 - typedef [basic_ostream](#)< [_CharT](#), [_Traits](#) > [ostream_type](#)

Public Member Functions

- [ostreambuf_iterator](#) ([ostream_type](#) &__s) throw ()
- [ostreambuf_iterator](#) ([streambuf_type](#) *__s) throw ()
- [ostreambuf_iterator](#) & [_M_put](#) (const [_CharT](#) *__ws, [streamsize](#) __len)
- bool [failed](#) () const throw ()
- [ostreambuf_iterator](#) & [operator*](#) ()
- [ostreambuf_iterator](#) & [operator++](#) ()
- [ostreambuf_iterator](#) & [operator++](#) (int)
- [ostreambuf_iterator](#) & [operator=](#) ([_CharT](#) __c)

Friends

- template<typename [_CharT2](#) >
[__gnu_cxx::__enable_if](#)< [__is_char](#)< [_CharT2](#) >::__value, [ostreambuf_iterator](#)< [_CharT2](#) >::__type **copy** ([istreambuf_iterator](#)< [_CharT2](#) >, [istreambuf_iterator](#)< [_CharT2](#) >, [ostreambuf_iterator](#)< [_CharT2](#) >)

5.599.1 Detailed Description

```
template<typename \_CharT, typename \_Traits> class std::ostreambuf\_iterator< \_CharT, \_Traits >
```

Provides output iterator semantics for streambufs.

Definition at line 204 of file [streambuf_iterator.h](#).

5.599.2 Member Typedef Documentation

5.599.2.1 `template<typename _CharT , typename _Traits > typedef _CharT
std::ostreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 211 of file ostreambuf_iterator.h.

5.599.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 114 of file stl_iterator_base_types.h.

5.599.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void
, void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 110 of file stl_iterator_base_types.h.

5.599.2.4 `template<typename _CharT , typename _Traits > typedef
basic_ostream<_CharT, _Traits> std::ostreambuf_iterator<
_CharT, _Traits >::ostream_type`

Public typedefs.

Definition at line 214 of file ostreambuf_iterator.h.

5.599.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 116 of file stl_iterator_base_types.h.

5.599.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file stl_iterator_base_types.h.

5.599.2.7 `template<typename _CharT, typename _Traits > typedef
basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator<
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 213 of file streambuf_iterator.h.

5.599.2.8 `template<typename _CharT, typename _Traits > typedef _Traits
std::ostreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 212 of file streambuf_iterator.h.

5.599.2.9 `typedef void std::iterator< output_iterator_tag, void, void, void,
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file stl_iterator_base_types.h.

5.599.3 Constructor & Destructor Documentation

5.599.3.1 `template<typename _CharT, typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
ostream_type & __s) throw () [inline]`

Construct output iterator from ostream.

Definition at line 229 of file streambuf_iterator.h.

5.599.3.2 `template<typename _CharT, typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
streambuf_type * __s) throw () [inline]`

Construct output iterator from streambuf.

Definition at line 233 of file streambuf_iterator.h.

5.599.4 Member Function Documentation

5.599.4.1 `template<typename _CharT, typename _Traits > bool
std::ostreambuf_iterator< _CharT, _Traits >::failed () const throw
() [inline]`

Return true if previous [operator=\(\)](#) failed.

Definition at line 263 of file `streambuf_iterator.h`.

5.599.4.2 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator* () [inline]`

Return `*this`.

Definition at line 248 of file `streambuf_iterator.h`.

5.599.4.3 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator++ () [inline]`

Return `*this`.

Definition at line 258 of file `streambuf_iterator.h`.

5.599.4.4 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator++ (int) [inline]`

Return `*this`.

Definition at line 253 of file `streambuf_iterator.h`.

5.599.4.5 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator= (_CharT __c) [inline]`

Write character to streambuf. Calls [streambuf.sputc\(\)](#).

Definition at line 238 of file `streambuf_iterator.h`.

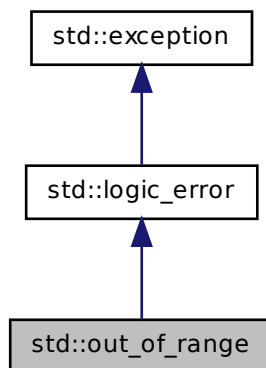
References `std::basic_streambuf< _CharT, _Traits >::sputc()`.

The documentation for this class was generated from the following file:

- [streambuf_iterator.h](#)

5.600 std::out_of_range Class Reference

Inheritance diagram for std::out_of_range:



Public Member Functions

- **out_of_range** (const [string](#) &__arg)
- virtual const char * **what** () const throw ()

5.600.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in [basic_string](#)).

Definition at line 96 of file `stdexcept`.

5.600.2 Member Function Documentation

5.600.2.1 virtual const char* std::logic_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.601 std::output_iterator_tag Struct Reference

Marking output iterators.

5.601.1 Detailed Description

Marking output iterators.

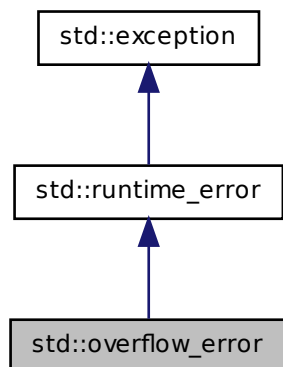
Definition at line 81 of file [stl_iterator_base_types.h](#).

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.602 std::overflow_error Class Reference

Inheritance diagram for std::overflow_error:



Public Member Functions

- **overflow_error** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.602.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 133 of file stdexcept.

5.602.2 Member Function Documentation

5.602.2.1 virtual const char* std::runtime_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.603 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Partial specialization of `owner_less` for [shared_ptr](#).

Inherits `_Sp_owner_less< shared_ptr< _Tp >, weak_ptr< _Tp > >`.

5.603.1 Detailed Description

`template<typename _Tp> struct std::owner_less< shared_ptr< _Tp > >`

Partial specialization of `owner_less` for [shared_ptr](#).

Definition at line 427 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

5.604 `std::owner_less< weak_ptr< _Tp > >` Struct Template Reference

Partial specialization of `owner_less` for [weak_ptr](#).

Inherits `_Sp_owner_less< weak_ptr< _Tp >, shared_ptr< _Tp > >`.

5.604.1 Detailed Description

`template<typename _Tp> struct std::owner_less< weak_ptr< _Tp > >`

Partial specialization of `owner_less` for [weak_ptr](#).

Definition at line 433 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

5.605 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference

`packaged_task`

Public Types

- `typedef _Res result_type`

Public Member Functions

- `template<typename _Fn > packaged_task (const _Fn &__fn)`
- `packaged_task (_Res(*__fn)(_ArgTypes...))`
- `packaged_task (packaged_task &&__other)`
- `template<typename _Fn > packaged_task (_Fn &&__fn)`
- `packaged_task (packaged_task &)`
- `future< _Res > get_future ()`
- `operator bool () const`
- `void operator() (_ArgTypes...__args)`
- `packaged_task & operator= (packaged_task &&__other)`
- `packaged_task & operator= (packaged_task &)`
- `void reset ()`
- `void swap (packaged_task &__other)`

5.605.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> class std::packaged_task< _Res(_ArgTypes...)>`

`packaged_task`

Definition at line 1183 of file `future`.

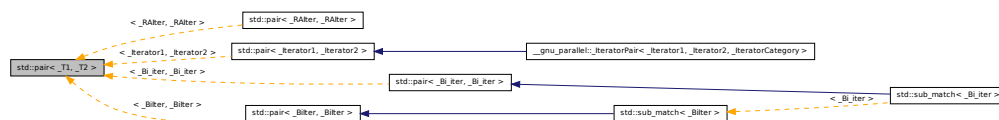
The documentation for this class was generated from the following file:

- `future`

5.606 std::pair< _T1, _T2 > Struct Template Reference

pair holds two objects of arbitrary type.

Inheritance diagram for std::pair< _T1, _T2 >:



Public Types

- typedef _T1 [first_type](#)
- typedef _T2 [second_type](#)

Public Member Functions

- [pair](#) ()
- [pair](#) (const _T1 &__a, const _T2 &__b)
- template<class _U1 , class = typename std::enable_if<std::is_convertible<_U1, _T1>::value>::type>
[pair](#) (_U1 &&__x, const _T2 &__y)
- template<class _U1 , class _U2 >
[pair](#) (const [pair](#)< _U1, _U2 > &__p)
- template<class _U1 , class _U2 >
[pair](#) ([pair](#)< _U1, _U2 > &&__p)
- template<class _U2 , class = typename std::enable_if<std::is_convertible<_U2, _T2>::value>::type>
[pair](#) (const _T1 &__x, _U2 &&__y)
- [pair](#) (const [pair](#) &)
- template<class... _Args1, class... _Args2>
[pair](#) (piecewise_construct_t, [tuple](#)< _Args1...> __first_args, [tuple](#)< _Args2...> __second_args)
- template<class _U1 , class _U2 , class = typename std::enable_if<std::is_convertible<_U1, _T1>::value && std::is_convertible<_U2, _T2>::value>::type>
[pair](#) (_U1 &&__x, _U2 &&__y)
- [pair](#) & [operator=](#) ([pair](#) &&__p)
- template<class _U1 , class _U2 >
[pair](#) & [operator=](#) (const [pair](#)< _U1, _U2 > &__p)

- `template<class _U1, class _U2 >`
`pair & operator= (pair< _U1, _U2 > &&__p)`
- `void swap (pair &__p)`

Public Attributes

- `_T1 first`
- `_T2 second`

5.606.1 Detailed Description

`template<class _T1, class _T2> struct std::pair< _T1, _T2 >`

`pair` holds two objects of arbitrary type.

Definition at line 81 of file `stl_pair.h`.

5.606.2 Member Typedef Documentation

5.606.2.1 `template<class _T1, class _T2> typedef _T1 std::pair< _T1, _T2 >::first_type`

`first_type` is the first bound type

Definition at line 83 of file `stl_pair.h`.

5.606.2.2 `template<class _T1, class _T2> typedef _T2 std::pair< _T1, _T2 >::second_type`

`second_type` is the second bound type

Definition at line 84 of file `stl_pair.h`.

5.606.3 Constructor & Destructor Documentation

5.606.3.1 `template<class _T1, class _T2> std::pair< _T1, _T2 >::pair ()`
`[inline]`

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 93 of file `stl_pair.h`.

5.607 `std::piecewise_constant_distribution<_RealType>` Class Template Reference 2901

5.606.3.2 `template<class _T1, class _T2> std::pair<_T1, _T2>::pair (const _T1 & __a, const _T2 & __b) [inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 97 of file `stl_pair.h`.

5.606.3.3 `template<class _T1, class _T2> template<class _U1, class _U2> std::pair<_T1, _T2>::pair (const pair<_U1, _U2> & __p) [inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 133 of file `stl_pair.h`.

5.606.4 Member Data Documentation

5.606.4.1 `template<class _T1, class _T2> _T1 std::pair<_T1, _T2>::first`

`first` is a copy of the first object

Definition at line 86 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, and `std::operator==()`.

5.606.4.2 `template<class _T1, class _T2> _T2 std::pair<_T1, _T2>::second`

`second` is a copy of the second object

Definition at line 87 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, and `std::operator==()`.

The documentation for this struct was generated from the following files:

- [stl_pair.h](#)
- [tuple](#)

5.607 `std::piecewise_constant_distribution<_RealType>` Class Template Reference _-

A [piecewise_constant_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef [_RealType](#) [result_type](#)

Public Member Functions

- template<typename [_InputIteratorB](#) , typename [_InputIteratorW](#) >
piecewise_constant_distribution ([_InputIteratorB](#) __bfirst, [_InputIteratorB](#) __bend, [_InputIteratorW](#) __wbegin)
- template<typename [_Func](#) >
piecewise_constant_distribution (size_t __nw, [_RealType](#) __xmin, [_RealType](#) __xmax, [_Func](#) __fw)
- **piecewise_constant_distribution** (const [param_type](#) &__p)
- template<typename [_Func](#) >
piecewise_constant_distribution ([initializer_list](#)< [_RealType](#) > __bl, [_Func](#) __fw)
- [std::vector](#)< double > [densities](#) () const
- [std::vector](#)< [_RealType](#) > [intervals](#) () const
- [result_type](#) max () const
- [result_type](#) min () const
- template<typename [_UniformRandomNumberGenerator](#) >
[result_type](#) **operator()** ([_UniformRandomNumberGenerator](#) &__urng, const [param_type](#) &__p)
- template<typename [_UniformRandomNumberGenerator](#) >
[result_type](#) **operator()** ([_UniformRandomNumberGenerator](#) &__urng)
- [param_type](#) [param](#) () const
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()

Friends

- template<typename [_RealType1](#) , typename [_CharT](#) , typename [_Traits](#) >
[std::basic_ostream](#)< [_CharT](#), [_Traits](#) > & **operator<<** ([std::basic_ostream](#)< [_CharT](#), [_Traits](#) > &, const [std::piecewise_constant_distribution](#)< [_RealType1](#) > &)
- template<typename [_RealType1](#) , typename [_CharT](#) , typename [_Traits](#) >
[std::basic_istream](#)< [_CharT](#), [_Traits](#) > & **operator>>** ([std::basic_istream](#)< [_CharT](#), [_Traits](#) > &, [std::piecewise_constant_distribution](#)< [_RealType1](#) > &)

5.607.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_constant_distribution<_RealType>`

A [piecewise_constant_distribution](#) random number distribution. The formula for the piecewise constant probability mass function is

Definition at line 4874 of file random.h.

5.607.2 Member Typedef Documentation

5.607.2.1 `template<typename _RealType = double> typedef _RealType
std::piecewise_constant_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4881 of file random.h.

5.607.3 Member Function Documentation

5.607.3.1 `template<typename _RealType = double> std::vector<double>
std::piecewise_constant_distribution<_RealType>::densities ()
const [inline]`

Returns a vector of the probability densities.

Definition at line 4973 of file random.h.

5.607.3.2 `template<typename _RealType = double> std::vector<_RealType>
std::piecewise_constant_distribution<_RealType>::intervals ()
const [inline]`

Returns a vector of the intervals.

Definition at line 4966 of file random.h.

5.607.3.3 `template<typename _RealType = double> result_type
std::piecewise_constant_distribution<_RealType>::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5002 of file random.h.

References `std::vector<_Tp, _Alloc>::back()`.

5.607.3.4 `template<typename _RealType = double> result_type
std::piecewise_constant_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4995 of file random.h.

References `std::vector< _Tp, _Alloc >::front()`.

5.607.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::piecewise_constant_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5010 of file random.h.

References `std::piecewise_constant_distribution< _RealType >::operator()()`, and `std::piecewise_constant_distribution< _RealType >::param()`.

Referenced by `std::piecewise_constant_distribution< _RealType >::operator()()`.

5.607.3.6 `template<typename _RealType = double> void
std::piecewise_constant_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4988 of file random.h.

5.607.3.7 `template<typename _RealType = double> param_type
std::piecewise_constant_distribution< _RealType >::param ()
const [inline]`

Returns the parameter set of the distribution.

Definition at line 4980 of file random.h.

Referenced by `std::piecewise_constant_distribution< _RealType >::operator()()`, and `std::operator==()`.

5.607.3.8 `template<typename _RealType = double> void
std::piecewise_constant_distribution<_RealType>::reset ()
[inline]`

Resets the distribution state.

Definition at line 4959 of file random.h.

5.607.4 Friends And Related Function Documentation

5.607.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream<_CharT, _Traits> & , const
std::piecewise_constant_distribution<_RealType1> &)
[friend]`

Inserts a `piecewise_constant_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `piecewise_constant_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.607.4.2 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits > std::basic_
istream<_CharT, _Traits>& operator>> (std::basic_istream<
_CharT, _Traits> & , std::piecewise_constant_distribution<
_RealType1> &) [friend]`

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `piecewise_constant_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.608 `std::piecewise_constant_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [piecewise_constant_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW >
param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- template<typename _Func >
param_type ([initializer_list](#)< _RealType > __bi, _Func __fw)
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< _RealType > **intervals** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)
- class [piecewise_constant_distribution](#)< _RealType >

5.608.1 Detailed Description

template<typename _RealType = double> struct std::piecewise_constant_distribution< _RealType >::param_type

Parameter type.

Definition at line 4883 of file [random.h](#).

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.609 `std::piecewise_linear_distribution<_RealType>` > Class Template Reference

A [`piecewise_linear_distribution`](#) random number distribution.

Classes

- struct [`param_type`](#)

Public Types

- typedef `_RealType` [`result_type`](#)

Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`
`piecewise_linear_distribution` (`_InputIteratorB __bfirst`, `_InputIteratorB __-`
`bend`, `_InputIteratorW __wbegin`)
- `template<typename _Func >`
`piecewise_linear_distribution` (`size_t __nw`, `_RealType __xmin`, `_RealType __-`
`xmax`, `_Func __fw`)
- **`piecewise_linear_distribution`** (`const` [`param_type`](#) `&__p`)
- `template<typename _Func >`
`piecewise_linear_distribution` (`initializer_list<_RealType> __bl`, `_Func __-`
`fw`)
- `std::vector<double>` [`densities`](#) () `const`
- `std::vector<_RealType>` [`intervals`](#) () `const`
- [`result_type`](#) `max` () `const`
- [`result_type`](#) `min` () `const`
- `template<typename _UniformRandomNumberGenerator >`
[`result_type`](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`, `const`
[`param_type`](#) `&__p`)
- `template<typename _UniformRandomNumberGenerator >`
[`result_type`](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- [`param_type`](#) `param` () `const`
- `void` [`param`](#) (`const` [`param_type`](#) `&__param`)
- `void` [`reset`](#) ()

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::piecewise_linear_distribution< _RealType1 > &)`
- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::piecewise_linear_distribution< _RealType1 > &)`

5.609.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_linear_distribution< _RealType >`

A [piecewise_linear_distribution](#) random number distribution. The formula for the piecewise linear probability mass function is

Definition at line 5082 of file random.h.

5.609.2 Member Typedef Documentation

5.609.2.1 `template<typename _RealType = double> typedef _RealType
std::piecewise_linear_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 5089 of file random.h.

5.609.3 Member Function Documentation

5.609.3.1 `template<typename _RealType = double> std::vector<double>
std::piecewise_linear_distribution< _RealType >::densities () const
[inline]`

Return a vector of the probability densities of the distribution.

Definition at line 5184 of file random.h.

5.609.3.2 `template<typename _RealType = double> std::vector<_RealType>
std::piecewise_linear_distribution< _RealType >::intervals () const
[inline]`

Return the intervals of the distribution.

Definition at line 5176 of file random.h.

5.609.3.3 `template<typename _RealType = double> result_type
std::piecewise_linear_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5213 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`.

5.609.3.4 `template<typename _RealType = double> result_type
std::piecewise_linear_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5206 of file random.h.

References `std::vector< _Tp, _Alloc >::front()`.

5.609.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::piecewise_linear_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5221 of file random.h.

References `std::piecewise_linear_distribution< _RealType >::operator()()`, and `std::piecewise_linear_distribution< _RealType >::param()`.

Referenced by `std::piecewise_linear_distribution< _RealType >::operator()()`.

5.609.3.6 `template<typename _RealType = double> void
std::piecewise_linear_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 5199 of file random.h.

5.609.3.7 `template<typename _RealType = double> param_type
std::piecewise_linear_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 5191 of file random.h.

Referenced by `std::piecewise_linear_distribution< _RealType >::operator()()`, and `std::operator==()`.

5.609.3.8 `template<typename _RealType = double> void
std::piecewise_linear_distribution< _RealType >::reset ()
[inline]`

Resets the distribution state.

Definition at line 5169 of file random.h.

5.609.4 Friends And Related Function Documentation

5.609.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::piecewise_linear_distribution< _RealType1 > &) [friend]`

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `piecewise_linear_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.610 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference 2911

5.609.4.2 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream<_CharT, _Traits> & ,
std::piecewise_linear_distribution<_RealType1> &) [friend]`

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `piecewise_linear_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.610 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `piecewise_linear_distribution<_RealType>` `distribution_type`

Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW>
param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)`
- `template<typename _Func>
param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)`
- `template<typename _Func>
param_type (initializer_list<_RealType> __bl, _Func __fw)`
- `std::vector<double> densities () const`
- `std::vector<_RealType> intervals () const`

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)
- class **piecewise_linear_distribution**< [_RealType](#) >

5.610.1 Detailed Description

template<typename [_RealType](#) = double> struct **std::piecewise_linear_distribution**< [_RealType](#) >::[param_type](#)

Parameter type.

Definition at line 5091 of file random.h.

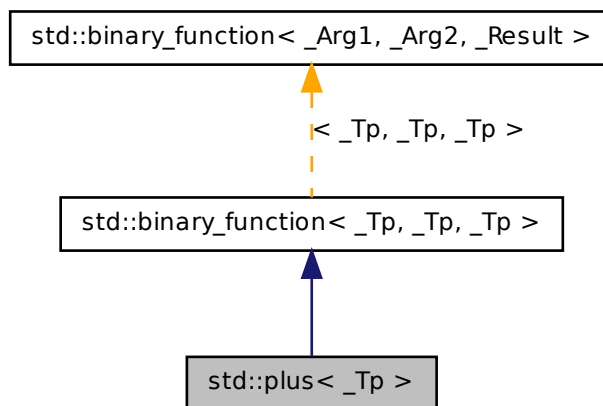
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.611 std::plus<_Tp> Struct Template Reference

One of the [math functors](#).

Inheritance diagram for `std::plus<_Tp>`:



Public Types

- typedef _Tp [first_argument_type](#)
- typedef _Tp [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

5.611.1 Detailed Description

`template<typename _Tp> struct std::plus< _Tp >`

One of the [math functors](#).

Definition at line 135 of file `stl_function.h`.

5.611.2 Member Typedef Documentation

5.611.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.611.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.611.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

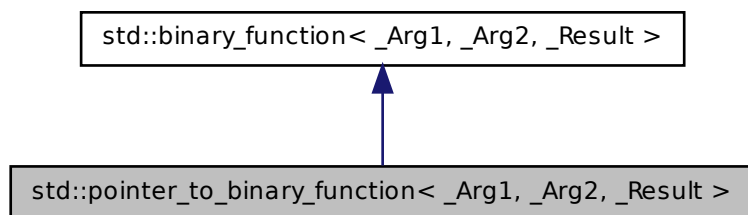
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.612 `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >` Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

Public Member Functions

- `pointer_to_binary_function` (`_Result`(*__x)(`_Arg1`, `_Arg2`))
- `_Result operator()` (`_Arg1` __x, `_Arg2` __y) const

Protected Attributes

- `_Result(* _M_ptr)` (`_Arg1`, `_Arg2`)

5.612.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> class
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 442 of file stl_function.h.

5.612.2 Member Typedef Documentation

5.612.2.1

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result
>::first_argument_type [inherited]
```

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.612.2.2

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result
>::result_type [inherited]
```

type of the return type

Definition at line 118 of file stl_function.h.

5.612.2.3

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result
>::second_argument_type [inherited]
```

the type of the second argument

Definition at line 117 of file stl_function.h.

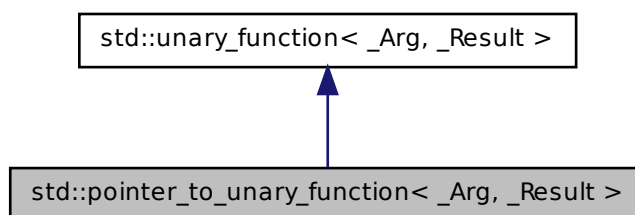
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.613 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_unary_function< _Arg, _Result >`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- `pointer_to_unary_function` (`_Result(*__x)(_Arg)`)
- `_Result operator()` (`_Arg __x`) const

Protected Attributes

- `_Result(* _M_ptr)(_Arg)`

5.613.1 Detailed Description

`template<typename _Arg, typename _Result> class std::pointer_to_unary_function< _Arg, _Result >`

One of the [adaptors for function pointers](#).

Definition at line 417 of file `stl_function.h`.

5.613.2 Member Typedef Documentation

5.613.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.613.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.614 `std::poisson_distribution< _IntType >` Class Template Reference

A discrete Poisson random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- `poisson_distribution` (double `__mean`=1.0)
- `poisson_distribution` (const [param_type](#) &`__p`)
- [result_type](#) `max` () const
- double `mean` () const
- [result_type](#) `min` () const

- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng, const`
`param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &, const std::poisson_distribution< _IntType1 > &)`
- `template<typename _IntType1 >`
`bool operator== (const std::poisson_distribution< _IntType1 > &__d1, const`
`std::poisson_distribution< _IntType1 > &__d2)`
- `template<typename _IntType1 , typename _CharT , typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::poisson_distribution< _IntType1 > &)`

5.614.1 Detailed Description

`template<typename _IntType = int> class std::poisson_distribution< _IntType`
`>`

A discrete Poisson random number distribution. The formula for the Poisson probability density function is $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$ where μ is the parameter of the distribution.

Definition at line 3973 of file random.h.

5.614.2 Member Typedef Documentation

5.614.2.1 `template<typename _IntType = int> typedef _IntType`
`std::poisson_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3980 of file random.h.

5.614.3 Member Function Documentation

5.614.3.1 `template<typename _IntType = int> result_type
std::poisson_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4067 of file random.h.

Referenced by std::poisson_distribution< _IntType >::operator()().

5.614.3.2 `template<typename _IntType = int> double
std::poisson_distribution< _IntType >::mean () const [inline]`

Returns the distribution parameter `mean`.

Definition at line 4038 of file random.h.

5.614.3.3 `template<typename _IntType = int> result_type
std::poisson_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4060 of file random.h.

5.614.3.4 `template<typename _IntType > template<typename
_UniformRandomNumberGenerator > poisson_distribution<
_IntType >::result_type std::poisson_distribution< _IntType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

A rejection algorithm when `mean >= 12` and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1192 of file random.tcc.

References `std::abs()`, `std::log()`, and `std::poisson_distribution< _IntType >::max()`.

5.614.3.5 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::poisson_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4075 of file random.h.

References `std::poisson_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::param()`.

Referenced by `std::poisson_distribution< _IntType >::operator()()`.

5.614.3.6 `template<typename _IntType = int> param_type
std::poisson_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4045 of file random.h.

Referenced by `std::poisson_distribution< _IntType >::operator()()`.

5.614.3.7 `template<typename _IntType = int> void std::poisson_distribution<
_IntType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 4053 of file random.h.

5.614.3.8 `template<typename _IntType = int> void std::poisson_distribution<
_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 4031 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

5.614.4 Friends And Related Function Documentation

5.614.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream<_CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & ,
const std::poisson_distribution< _IntType1 > &) [friend]`

Inserts a poisson_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A poisson_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.614.4.2 `template<typename _IntType = int> template<typename _IntType1
> bool operator== (const std::poisson_distribution< _IntType1 >
& __d1, const std::poisson_distribution< _IntType1 > & __d2)
[friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4090 of file random.h.

5.614.4.3 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream< _CharT, _Traits > & ,
std::poisson_distribution< _IntType1 > &) [friend]`

Extracts a poisson_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A poisson_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.615 `std::poisson_distribution< _IntType >::param_type` Struct Reference

Public Types

- typedef [poisson_distribution](#)< _IntType > `distribution_type`

Public Member Functions

- `param_type` (double __mean=1.0)
- double `mean` () const

Friends

- bool `operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)
- class `poisson_distribution`< _IntType >

5.615.1 Detailed Description

`template<typename _IntType = int> struct std::poisson_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3982 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.616 `std::priority_queue< _Tp, _Sequence, _Compare >` Class Template Reference

A standard container automatically sorting its contents.

Public Types

- `typedef _Sequence::const_reference` **const_reference**
- `typedef _Sequence` **container_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size_type**
- `typedef _Sequence::value_type` **value_type**

Public Member Functions

- [priority_queue](#) (const _Compare &__x, const _Sequence &__s)
- **priority_queue** (const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence())
- `template<typename _InputIterator >`
priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence())
- **priority_queue** ([priority_queue](#) &&__pq)
- `template<typename _InputIterator >`
[priority_queue](#) (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s)
- `template<typename... _Args>`
void **emplace** (_Args &&...__args)
- bool [empty](#) () const
- [priority_queue](#) & **operator=** ([priority_queue](#) &&__pq)
- void [pop](#) ()
- void [push](#) (const value_type &__x)
- void **push** (value_type &&__x)
- size_type [size](#) () const
- void **swap** ([priority_queue](#) &__pq)
- const_reference [top](#) () const

Protected Attributes

- _Sequence **c**
- _Compare **comp**

5.616.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _  
Compare = less<typename _Sequence::value_type>> class std::priority_queue<  
_Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents. This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that con-

tainer. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

Note

No equality/comparison operators are provided for `priority_queue`. Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 359 of file `stl_queue.h`.

5.616.2 Constructor & Destructor Documentation

5.616.2.1 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue(
const _Compare & __x, const _Sequence & __s) [inline,
explicit]`

Default constructor creates no elements.

Definition at line 394 of file `stl_queue.h`.

References `std::make_heap()`.

5.616.2.2 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
template<typename _InputIterator > std::priority_queue< _Tp,
_Sequence, _Compare >::priority_queue (_InputIterator __first,
_InputIterator __last, const _Compare & __x, const _Sequence &
__s) [inline]`

Builds a queue from a range.

Parameters

first An input iterator.

last An input iterator.

x A comparison functor describing a strict weak ordering.

s An initial sequence with which to start.

Begins by copying *s*, inserting a copy of the elements from [first,last) into the copy of *s*, then ordering the copy according to *x*.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 434 of file stl_queue.h.

References std::make_heap().

5.616.3 Member Function Documentation

5.616.3.1 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
bool std::priority_queue< _Tp, _Sequence, _Compare >::empty ()
const [inline]`

Returns true if the queue is empty.

Definition at line 471 of file stl_queue.h.

Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

5.616.3.2 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
void std::priority_queue< _Tp, _Sequence, _Compare >::pop ()
[inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 534 of file `stl_queue.h`.

References `std::pop_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

```
5.616.3.3  template<typename _Tp, typename _Sequence = vector<_Tp>,
              typename _Compare = less<typename _Sequence::value_type>>
              void std::priority_queue<_Tp, _Sequence, _Compare >::push (
                const value_type & __x ) [inline]
```

Add data to the queue.

Parameters

x Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 499 of file `stl_queue.h`.

References `std::push_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

```
5.616.3.4  template<typename _Tp, typename _Sequence = vector<_Tp>,
              typename _Compare = less<typename _Sequence::value_type>>
              size_type std::priority_queue<_Tp, _Sequence, _Compare >::size (
                ) const [inline]
```

Returns the number of elements in the queue.

Definition at line 476 of file `stl_queue.h`.

```
5.616.3.5  template<typename _Tp, typename _Sequence = vector<_Tp>,
              typename _Compare = less<typename _Sequence::value_type>>
              const_reference std::priority_queue<_Tp, _Sequence, _Compare
                >::top ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 484 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl_queue.h](#)

5.617 `std::promise< _Res >` Class Template Reference

Primary template for `promise`.

Public Member Functions

- `promise` ([promise](#) &&__rhs)
- `promise` (const [promise](#) &)
- `future< _Res > get_future ()`
- `promise & operator=` ([promise](#) &&__rhs)
- `promise & operator=` (const [promise](#) &)
- void `set_exception` (exception_ptr __p)
- void `set_value` (const _Res &__r)
- void `set_value` (_Res &&__r)
- void `swap` ([promise](#) &__rhs)

Friends

- class `_State::Setter`

5.617.1 Detailed Description

`template<typename _Res> class std::promise< _Res >`

Primary template for `promise`.

Definition at line 825 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

5.618 `std::promise< _Res & >` Class Template Reference

Partial specialization for `promise<R&>`

Public Member Functions

- `promise` (`promise` &&__rhs)
- `promise` (const `promise` &)
- `future< _Res & > get_future` ()
- `promise` & `operator=` (`promise` &&__rhs)
- `promise` & `operator=` (const `promise` &)
- void `set_exception` (exception_ptr __p)
- void `set_value` (_Res &__r)
- void `swap` (`promise` &__rhs)

Friends

- class `_State::_Setter`

5.618.1 Detailed Description

`template<typename _Res> class std::promise< _Res & >`

Partial specialization for `promise<R&>`

Definition at line 915 of file `future`.

The documentation for this class was generated from the following file:

- `future`

5.619 `std::promise< void >` Class Template Reference

Explicit specialization for `promise<void>`

Public Member Functions

- `promise` (`promise` &&__rhs)
- `promise` (const `promise` &)

- `future< void > get_future ()`
- `promise & operator= (promise &&__rhs)`
- `promise & operator= (const promise &)`
- `void set_exception (exception_ptr __p)`
- `void set_value ()`
- `void swap (promise &&__rhs)`

Friends

- class `_State::_Setter`

5.619.1 Detailed Description

`template<> class std::promise< void >`

Explicit specialization for `promise<void>`

Definition at line 993 of file `future`.

The documentation for this class was generated from the following file:

- `future`

5.620 `std::queue< _Tp, _Sequence >` Class Template Reference

A standard container giving FIFO behavior.

Public Types

- `typedef _Sequence::const_reference const_reference`
- `typedef _Sequence container_type`
- `typedef _Sequence::reference reference`
- `typedef _Sequence::size_type size_type`
- `typedef _Sequence::value_type value_type`

Public Member Functions

- `queue (const _Sequence &__c)`
- `queue (_Sequence &&__c=_Sequence())`

- **queue** ([queue](#) &&__q)
- reference [back](#) ()
- const_reference [back](#) () const
- template<typename... _Args>
void **emplace** (_Args &&...__args)
- bool [empty](#) () const
- const_reference [front](#) () const
- reference [front](#) ()
- [queue](#) & **operator=** ([queue](#) &&__q)
- void [pop](#) ()
- void **push** (value_type &&__x)
- void [push](#) (const value_type &__x)
- size_type [size](#) () const
- void **swap** ([queue](#) &__q)

Protected Attributes

- _Sequence [c](#)

Friends

- template<typename _Tp1, typename _Seq1 >
bool **operator**< (const [queue](#)< _Tp1, _Seq1 > &, const [queue](#)< _Tp1, _Seq1 > &)
- template<typename _Tp1, typename _Seq1 >
bool **operator==** (const [queue](#)< _Tp1, _Seq1 > &, const [queue](#)< _Tp1, _Seq1 > &)

5.620.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>> class std::queue<_Tp, _Sequence >

A standard container giving FIFO behavior. Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to [std::deque](#), but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as [std::list](#) or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 89 of file `stl_queue.h`.

5.620.2 Constructor & Destructor Documentation

5.620.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
std::queue<_Tp, _Sequence>::queue (const _Sequence & __c)
[inline, explicit]`

Default constructor creates no elements.

Definition at line 134 of file `stl_queue.h`.

5.620.3 Member Function Documentation

5.620.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::back () [inline]`

Returns a read/write reference to the data at the last element of the queue.

Definition at line 191 of file `stl_queue.h`.

5.620.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::back () const
[inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 202 of file `stl_queue.h`.

5.620.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue<_Tp, _Sequence>::empty () const [inline]`

Returns true if the queue is empty.

Definition at line 156 of file `stl_queue.h`.

5.620.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::front () [inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 169 of file `stl_queue.h`.

5.620.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 180 of file `stl_queue.h`.

5.620.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::pop () [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 244 of file `stl_queue.h`.

5.620.3.7 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::push (const value_type & __x
) [inline]`

Add data to the end of the queue.

Parameters

x Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 218 of file `stl_queue.h`.

5.620.3.8 `template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size () const [inline]`

Returns the number of elements in the queue.

Definition at line 161 of file `stl_queue.h`.

5.620.4 Member Data Documentation

5.620.4.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequence std::queue<_Tp, _Sequence>::c [protected]`

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 122 of file `stl_queue.h`.

Referenced by `std::operator==()`.

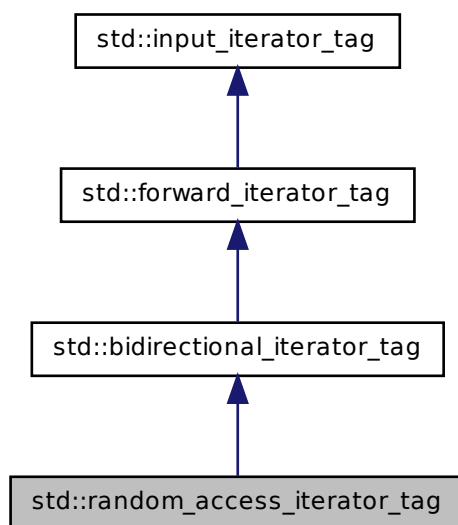
The documentation for this class was generated from the following file:

- [stl_queue.h](#)

5.621 `std::random_access_iterator_tag` Struct Reference

Random-access iterators support a superset of bidirectional /// iterator operations.

Inheritance diagram for `std::random_access_iterator_tag`:



5.621.1 Detailed Description

Random-access iterators support a superset of bidirectional /// iterator operations.

Definition at line 92 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.622 `std::random_device` Class Reference

Public Types

- typedef unsigned int [result_type](#)

Public Member Functions

- **random_device** (const [std::string](#) &__token="mt19937")
- **random_device** (const [random_device](#) &)
- double **entropy** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- [result_type](#) **operator**() ()
- void **operator=** (const [random_device](#) &)

5.622.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1521 of file random.h.

5.622.2 Member Typedef Documentation

5.622.2.1 typedef unsigned int std::random_device::result_type

The type of the generated random value.

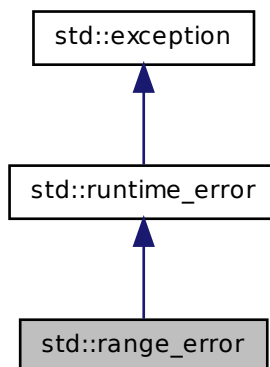
Definition at line 1525 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

5.623 std::range_error Class Reference

Inheritance diagram for std::range_error:



Public Member Functions

- **range_error** (const [string](#) &__arg)
- virtual const char * **what** () const throw ()

5.623.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 126 of file `stdexcept`.

5.623.2 Member Function Documentation

5.623.2.1 virtual const char* std::runtime_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.624 `std::ratio< _Num, _Den >` Struct Template Reference

Provides compile-time rational arithmetic.

Public Member Functions

- **static_assert** (`_Den!=0`, "denominator cannot be zero")
- **static_assert** (`_Num >= __INTMAX_MAX__ && _Den >= __INTMAX_MAX__`, "out of range")

Static Public Attributes

- static const `intmax_t` **den**
- static const `intmax_t` **num**

5.624.1 Detailed Description

`template<intmax_t _Num, intmax_t _Den = 1> struct std::ratio< _Num, _Den >`

Provides compile-time rational arithmetic. This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type `intmax_t`. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

Definition at line 151 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.625 `std::ratio_add< _R1, _R2 >` Struct Template Reference

[ratio_add](#)

Public Types

- typedef [ratio](#)< `__safe_add< __safe_multiply< _R1::num,(_R2::den/__gcd)>::value, __safe_multiply< _R2::num,(_R1::den/__gcd)>::value >::value, __safe_multiply< _R1::den,(_R2::den/__gcd)>::value >` **type**

5.625.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_add< _R1, _R2 >
```

[ratio_add](#)

Definition at line 173 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.626 `std::ratio_divide< _R1, _R2 >` Struct Template Reference

[ratio_divide](#)

Public Types

- typedef [ratio_multiply](#)< `_R1, ratio< _R2::den, _R2::num > >::type` **type**

Public Member Functions

- `static_assert` (`_R2::num!=0,"division by 0"`)

5.626.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_divide< _R1, _R2 >
```

[ratio_divide](#)

Definition at line 216 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.627 `std::ratio_equal< _R1, _R2 >` Struct Template Reference

[ratio_equal](#)

Inherits `integral_constant< bool, _R1::num==_R2::num &&_R1::den==_R2::den >`.

5.627.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_equal< _R1, _R2 >
```

[ratio_equal](#)

Definition at line 227 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.628 `std::ratio_greater< _R1, _R2 >` Struct Template Reference

[ratio_greater](#)

Inherits `integral_constant< bool, ratio_less< _R2, _R1 >::value >`.

5.628.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_greater< _R1, _R2 >
```

[ratio_greater](#)

Definition at line 269 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.629 `std::ratio_greater_equal< _R1, _R2 >` Struct Template Reference

[ratio_greater_equal](#)

Inherits `integral_constant< bool, !ratio_less< _R1, _R2 >::value >`.

5.629.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_greater_equal< _R1,
_R2 >
```

[ratio_greater_equal](#)

Definition at line 275 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.630 `std::ratio_less< _R1, _R2 >` Struct Template Reference

[ratio_less](#)

Inherits `__ratio_less_impl::type< _R1, _R2 >`.

5.630.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_less< _R1, _R2 >
```

[ratio_less](#)

Definition at line 257 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.631 `std::ratio_less_equal< _R1, _R2 >` Struct Template Reference

[ratio_less_equal](#)

Inherits `integral_constant< bool, !ratio_less< _R2, _R1 >::value >`.

5.631.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_less_equal< _R1, _R2
>
```

[ratio_less_equal](#)

Definition at line 263 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.632 `std::ratio_multiply< _R1, _R2 >` Struct Template Reference

[ratio_multiply](#)

Public Types

- typedef [ratio](#)< `__safe_multiply<(_R1::num/ __gcd1),(_R2::num/ __gcd2)>::value,` `__safe_multiply<(_R1::den/ __gcd2),(_R2::den/ __gcd1)>::value` > **type**

5.632.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_multiply< _R1, _R2
>
```

[ratio_multiply](#)

Definition at line 198 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.633 `std::ratio_not_equal< _R1, _R2 >` Struct Template Reference

[ratio_not_equal](#)

Inherits `integral_constant< bool, !ratio_equal< _R1, _R2 >::value >`.

5.633.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_not_equal< _R1, _R2
>
```

[ratio_not_equal](#)

Definition at line 233 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.634 `std::ratio_subtract< _R1, _R2 >` Struct Template Reference

[ratio_subtract](#)

Public Types

- typedef [ratio_add](#)< _R1, [ratio](#)<-_R2::num, _R2::den > >::type type

5.634.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_subtract< _R1, _R2
>
```

[ratio_subtract](#)

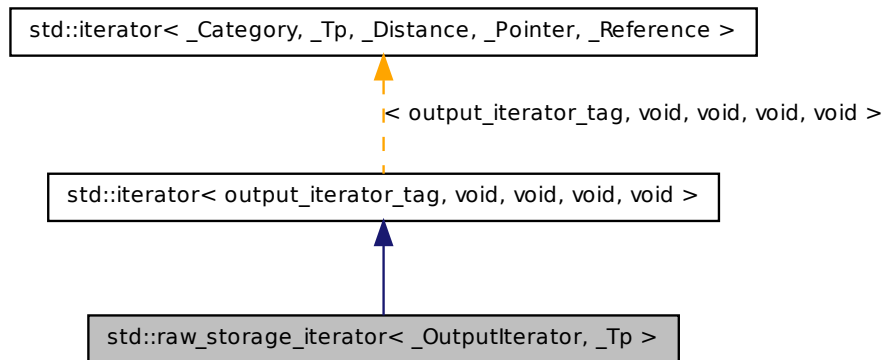
Definition at line 189 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.635 `std::raw_storage_iterator< _OutputIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::raw_storage_iterator< _OutputIterator, _Tp >`:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

Public Member Functions

- **`raw_storage_iterator`** (`_OutputIterator __x`)
- **`raw_storage_iterator`** & **`operator*`** ()
- **`raw_storage_iterator`** < `_OutputIterator, _Tp` > & **`operator++`** ()
- **`raw_storage_iterator`** < `_OutputIterator, _Tp` > **`operator++`** (int)
- **`raw_storage_iterator`** & **`operator=`** (const `_Tp` &__element)

Protected Attributes

- `_OutputIterator _M_iter`

5.635.1 Detailed Description

`template<class _OutputIterator, class _Tp> class std::raw_storage_iterator< _OutputIterator, _Tp >`

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 67 of file `stl_raw_storage_iter.h`.

5.635.2 Member Typedef Documentation

5.635.2.1 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 114 of file `stl_iterator_base_types.h`.

5.635.2.2 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 110 of file `stl_iterator_base_types.h`.

5.635.2.3 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 116 of file `stl_iterator_base_types.h`.

5.635.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 118 of file `stl_iterator_base_types.h`.

5.635.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file stl_iterator_base_types.h.

The documentation for this class was generated from the following file:

- [stl_raw_storage_iter.h](#)

5.636 std::recursive_mutex Class Reference

[recursive_mutex](#)

Public Types

- `typedef __native_type * native_handle_type`

Public Member Functions

- `recursive_mutex` (const [recursive_mutex](#) &)
- `void lock ()`
- `native_handle_type native_handle ()`
- `recursive_mutex & operator= (const recursive_mutex &)`
- `bool try_lock ()`
- `void unlock ()`

5.636.1 Detailed Description

[recursive_mutex](#)

Definition at line 114 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

5.637 std::recursive_timed_mutex Class Reference

[recursive_timed_mutex](#)

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **recursive_timed_mutex** (const [recursive_timed_mutex](#) &)
- void **lock** ()
- native_handle_type **native_handle** ()
- [recursive_timed_mutex](#) & **operator=** (const [recursive_timed_mutex](#) &)
- bool **try_lock** ()
- template<class _Rep, class _Period >
bool **try_lock_for** (const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<class _Clock, class _Duration >
bool **try_lock_until** (const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- void **unlock** ()

5.637.1 Detailed Description

[recursive_timed_mutex](#)

Definition at line 270 of file mutex.

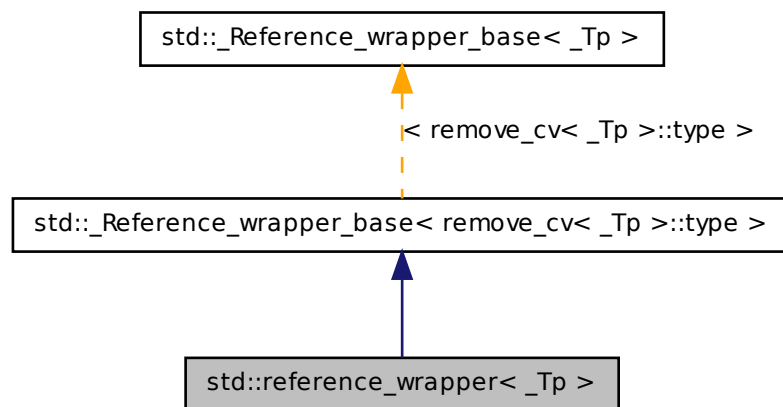
The documentation for this class was generated from the following file:

- [mutex](#)

5.638 [std::reference_wrapper](#)< _Tp > Class Template Reference

Primary class template for [reference_wrapper](#).

Inheritance diagram for std::reference_wrapper< _Tp >:



Public Types

- `typedef _Tp type`

Public Member Functions

- `reference_wrapper (_Tp &__indata)`
- `reference_wrapper (_Tp &&)`
- `reference_wrapper (const reference_wrapper< _Tp > &__inref)`
- `_Tp & get () const`
- `operator _Tp & () const`
- `template<typename... _Args>
result_of< _M_func_type(_Args...)>::type operator() (_Args &&...__args)
const`
- `reference_wrapper & operator= (const reference_wrapper< _Tp > &__inref)`

5.638.1 Detailed Description

`template<typename _Tp> class std::reference_wrapper< _Tp >`

Primary class template for [reference_wrapper](#).

Definition at line 381 of file functional.

The documentation for this class was generated from the following file:

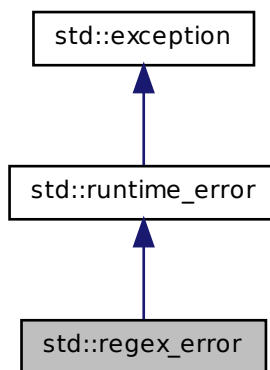
- [functional](#)

5.639 std::regex_error Class Reference

A regular expression exception class.

The regular expression library throws objects of this class on error.

Inheritance diagram for std::regex_error:



Public Member Functions

- [regex_error](#) ([regex_constants::error_type](#) __ecode)
- [regex_constants::error_type](#) code () const
- virtual const char * [what](#) () const throw ()

Protected Attributes

- `regex_constants::error_type _M_code`

5.639.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 127 of file `regex_error.h`.

5.639.2 Constructor & Destructor Documentation

5.639.2.1 `std::regex_error::regex_error (regex_constants::error_type __ecode) [inline, explicit]`

Constructs a `regex_error` object.

Parameters

ecode the regex error code.

Definition at line 137 of file `regex_error.h`.

5.639.3 Member Function Documentation

5.639.3.1 `regex_constants::error_type std::regex_error::code () const [inline]`

Gets the regex error code.

Returns

the regex error code.

Definition at line 147 of file `regex_error.h`.

5.639.3.2 `virtual const char* std::runtime_error::what () const throw () [virtual, inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [regex_error.h](#)

5.640 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef const [value_type](#) * **pointer**
- typedef const [value_type](#) & **reference**
- typedef [basic_regex](#)< _Ch_type, _Rx_traits > **regex_type**
- typedef [match_results](#)< _Bi_iter > **value_type**

Public Member Functions

- [regex_iterator](#) ()
- [regex_iterator](#) (_Bi_iter __a, _Bi_iter __b, const [regex_type](#) &__re, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- [regex_iterator](#) (const [regex_iterator](#) &__rhs)
- bool [operator!=](#) (const [regex_iterator](#) &__rhs)
- const [value_type](#) & [operator*](#) ()
- [regex_iterator](#) & [operator++](#) ()
- [regex_iterator](#) [operator++](#) (int)
- const [value_type](#) * [operator->](#) ()
- [regex_iterator](#) & [operator=](#) (const [regex_iterator](#) &__rhs)
- bool [operator==](#) (const [regex_iterator](#) &__rhs)

5.640.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> class
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2149 of file `regex.h`.

5.640.2 Constructor & Destructor Documentation

5.640.2.1 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_iterator ()`

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.640.2.2 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_iterator (_Bi_iter __a, _Bi_iter __b, const
regex_type & __re, regex_constants::match_flag_type __m =
regex_constants::match_default)`

Constructs a regex_iterator...

Parameters

a [IN] The start of a text range to search.

b [IN] One-past-the-end of the text range to search.

re [IN] The regular expression to match.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.640.2.3 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_iterator (const regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits > & __rhs)`

Copy constructs a regex_iterator.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.640.3 Member Function Documentation

5.640.3.1 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator!=(const regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits > & __rhs)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.640.3.2 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> const value_type& std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator* ()`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.640.3.3 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> regex_iterator std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator++ (int)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.640 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference **2953**

5.640.3.4 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator++ ()`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.640.3.5 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> const value_type* std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator-> ()`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.640.3.6 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator= (const regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.640.3.7 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator== (const regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

The documentation for this class was generated from the following file:

- [regex.h](#)

5.641 `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef const [value_type](#) * **pointer**
- typedef const [value_type](#) & **reference**
- typedef [basic_regex< _Ch_type, _Rx_traits >](#) **regex_type**
- typedef [sub_match< _Bi_iter >](#) **value_type**

Public Member Functions

- [regex_token_iterator](#) ()
- [regex_token_iterator](#) ([_Bi_iter](#) __a, [_Bi_iter](#) __b, const [regex_type](#) &__re, int __submatch=0, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- template<[std::size_t](#) _Nm>
[regex_token_iterator](#) ([_Bi_iter](#) __a, [_Bi_iter](#) __b, const [regex_type](#) &__re, const int(&__submatches)[_Nm], [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- [regex_token_iterator](#) (const [regex_token_iterator](#) &__rhs)
- [regex_token_iterator](#) ([_Bi_iter](#) __a, [_Bi_iter](#) __b, const [regex_type](#) &__re, const [std::vector< int >](#) &__submatches, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- bool [operator!=](#) (const [regex_token_iterator](#) &__rhs)
- const [value_type](#) & [operator*](#) ()
- [regex_token_iterator](#) & [operator++](#) ()
- [regex_token_iterator](#) [operator++](#) (int)
- const [value_type](#) * [operator->](#) ()
- [regex_token_iterator](#) & [operator=](#) (const [regex_token_iterator](#) &__rhs)
- bool [operator==](#) (const [regex_token_iterator](#) &__rhs)

5.641.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> class
std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a [std::sub_match](#) object.

Definition at line 2264 of file `regex.h`.

5.641.2 Constructor & Destructor Documentation

5.641.2.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator ()`

Default constructs a `regex_token_iterator`.

Todo

Implement this function.

A default-constructed `regex_token_iterator` is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

5.641.2.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type & __re, int __submatch = 0, regex_constants::match_flag_type __m = regex_constants::match_default)`

Constructs a `regex_token_iterator`...

Parameters

- a* [IN] The start of the text to search.
- b* [IN] One-past-the-end of the text to search.
- re* [IN] The regular expression to search for.

submatch [IN] Which submatch to return. There are some special values for this parameter:

- -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)
- 0 the entire string matching the subexpression is returned for each match within the text.
- >0 enumerates only the indicated subexpression from a match within the text.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

```
5.641.2.3 template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::regex_token_iterator ( _Bi_iter __a,
_Bi_iter __b, const regex_type & __re, const std::vector< int
> & __submatches, regex_constants::match_flag_type __m =
regex_constants::match_default )
```

Constructs a `regex_token_iterator`...

Parameters

a [IN] The start of the text to search.

b [IN] One-past-the-end of the text to search.

re [IN] The regular expression to search for.

submatches [IN] A list of subexpressions to return for each regular expression match within the text.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.641.2.4 `template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> template<std::size_t
_Nm> std::regex_token_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_token_iterator (_Bi_iter __a,
_Bi_iter __b, const regex_type & __re, const int(&)
__submatches[_Nm], regex_constants::match_flag_type __m =
regex_constants::match_default)`

Constructs a regex_token_iterator...

Parameters

a [IN] The start of the text to search.

b [IN] One-past-the-end of the text to search.

re [IN] The regular expression to search for.

submatches [IN] A list of subexpressions to return for each regular expression match within the text.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.641.2.5 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (const
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Copy constructs a regex_token_iterator.

Parameters

rhs [IN] A regex_token_iterator to copy.

Todo

Implement this function.

5.641.3 Member Function Documentation

5.641.3.1 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator!= (const regex_token_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Compares a `regex_token_iterator` to another for inequality.

Todo

Implement this function.

5.641.3.2 `template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> const value_type&
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator* ()`

Dereferences a `regex_token_iterator`.

Todo

Implement this function.

5.641.3.3 `template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator++ (int)`

Postincrements a `regex_token_iterator`.

Todo

Implement this function.

5.641.3.4 `template<typename _Bi_iter, typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator&
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator++ ()`

Increments a regex_token_iterator.

Todo

Implement this function.

5.641.3.5 `template<typename _Bi_iter, typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> const value_type*
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator-> ()`

Selects a regex_token_iterator member.

Todo

Implement this function.

5.641.3.6 `template<typename _Bi_iter, typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator&
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator= (const regex_token_iterator< _Bi_iter, _Ch_type,
_Rx_traits > & __rhs)`

Assigns a regex_token_iterator to another.

Parameters

rhs [IN] A regex_token_iterator to copy.

Todo

Implement this function.

5.641.3.7 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator==(const regex_token_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Compares a `regex_token_iterator` to another for equality.

Todo

Implement this function.

The documentation for this class was generated from the following file:

- [regex.h](#)

5.642 `std::regex_traits< _Ch_type >` Struct Template Reference

Describes aspects of a regular expression.

Public Types

- `typedef std::ctype_base::mask char_class_type`
- `typedef _Ch_type char_type`
- `typedef std::locale locale_type`
- `typedef std::basic_string< char_type > string_type`

Public Member Functions

- [regex_traits](#) ()
- [locale_type](#) `getloc` () const
- [locale_type](#) `imbue` ([locale_type](#) __loc)
- `bool isctype` (_Ch_type __c, char_class_type __f) const
- `template<typename _Fwd_iter >`
`char_class_type lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __`
`__icase=false) const`
- `template<typename _Fwd_iter >`
`string_type lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const`
- `template<typename _Fwd_iter >`
`string_type transform (_Fwd_iter __first, _Fwd_iter __last) const`

- template<typename _Fwd_iter >
 [string_type transform_primary](#) (_Fwd_iter __first, _Fwd_iter __last) const
- char_type [translate](#) (char_type __c) const
- char_type [translate_nocase](#) (char_type __c) const
- int [value](#) (_Ch_type __ch, int __radix) const

Static Public Member Functions

- static std::size_t [length](#) (const char_type *__p)

Protected Attributes

- [locale_type _M_locale](#)

5.642.1 Detailed Description

template<typename _Ch_type> struct std::regex_traits< _Ch_type >

Describes aspects of a regular expression. A regular expression traits class that satisfies the requirements of section [28.7].

The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 52 of file regex.h.

5.642.2 Constructor & Destructor Documentation

5.642.2.1 **template<typename _Ch_type > std::regex_traits< _Ch_type >::regex_traits () [inline]**

Constructs a default traits object.

Definition at line 64 of file regex.h.

5.642.3 Member Function Documentation

5.642.3.1 **template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::getloc () const [inline]**

Gets a copy of the current locale in use by the [regex_traits](#) object.

Definition at line 273 of file regex.h.

5.642.3.2 `template<typename _Ch_type > locale_type std::regex_traits<_Ch_type >::imbue (locale_type __loc) [inline]`

Imbues the [regex_traits](#) object with a copy of a new locale.

Parameters

loc A locale.

Returns

a copy of the previous locale in use by the [regex_traits](#) object.

Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by *this.

Definition at line 262 of file regex.h.

References `std::swap()`.

5.642.3.3 `template<typename _Ch_type > static std::size_t std::regex_traits<_Ch_type >::length (const char_type * __p) [inline, static]`

Gives the length of a C-style string starting at `__p`.

Parameters

`__p` a pointer to the start of a character sequence.

Returns

the number of characters between `*__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 78 of file regex.h.

References `std::basic_string<_CharT, _Traits, _Alloc >::length()`.

5.642.3.4 `template<typename _Ch_type > template<typename _Fwd_iter > char_class_type std::regex_traits<_Ch_type >::lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase = false) const [inline]`

Maps one or more characters to a named character classification.

Parameters

- first* beginning of the character sequence.
- last* one-past-the-end of the character sequence.
- icase* ignores the case of the classification name.

Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[first, last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependant classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`
- `alnum`
- `alpha`
- `blank`
- `cntrl`
- `digit`
- `graph`
- `lower`
- `print`
- `punct`
- `space`
- `upper`
- `xdigit`

Todo

Implement this function.

Definition at line 218 of file `regex.h`.

Referenced by `std::regex_traits<_Ch_type>::isctype()`.

5.642.3.5 `template<typename _Ch_type > template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::lookup_collatename (
_Fwd_iter __first, _Fwd_iter __last) const [inline]`

Gets a collation element by name.

Parameters

first beginning of the collation element name.

last one-past-the-end of the collation element name.

Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [first, last). Returns an empty string if the character sequence is not a valid collating element.

Todo

Implement this function.

Definition at line 175 of file regex.h.

5.642.3.6 `template<typename _Ch_type > template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::transform (_Fwd_iter
__first, _Fwd_iter __last) const [inline]`

Gets a sort key for a character sequence.

Parameters

first beginning of the character sequence.

last one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

Returns

a locale-specific sort key equivalent to the input range.

Exceptions

std::bad_cast if the current locale does not have a collate facet.

Definition at line 131 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Traits, _Alloc >::size(), std::collate< _CharT >::transform(), and std::use_facet().

5.642.3.7 `template<typename _Ch_type > template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::transform_primary (
_Fwd_iter __first, _Fwd_iter __last) const [inline]`

Gets a sort key for a character sequence, independant of case.

Parameters

first beginning of the character sequence.

last one-past-the-end of the character sequence.

Effects: if typeid(use_facet<collate<_Ch_type> >) == typeid(collate_byname<_Ch_type>) and the form of the sort key returned by collate_byname<_Ch_type>::transform(first, last) is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

Todo

Implement this function.

Definition at line 157 of file regex.h.

5.642.3.8 `template<typename _Ch_type > char_type std::regex_traits<
_Ch_type >::translate (char_type __c) const [inline]`

Performs the identity translation.

Parameters

c A character to the locale-specific character set.

Returns

c.

Definition at line 89 of file regex.h.

5.642.3.9 `template<typename _Ch_type > char_type std::regex_traits<_Ch_type >::translate_nocase (char_type __c) const [inline]`

Translates a character into a case-insensitive equivalent.

Parameters

c A character to the locale-specific character set.

Returns

the locale-specific lower-case equivalent of *c*.

Exceptions

[*std::bad_cast*](#) if the imbued locale does not support the ctype facet.

Definition at line 102 of file `regex.h`.

References `std::tolower()`.

The documentation for this struct was generated from the following file:

- [regex.h](#)

5.643 `std::remove_reference< _Tp >` Struct Template Reference

[remove_reference](#)

Public Types

- `typedef _Tp type`

5.643.1 Detailed Description

`template<typename _Tp> struct std::remove_reference< _Tp >`

[remove_reference](#)

Definition at line 98 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.644 std::reverse_iterator< _Iterator > Class Template Reference

Inheritance diagram for std::reverse_iterator< _Iterator >:



Public Types

- typedef __traits_type::difference_type [difference_type](#)
- typedef [iterator_traits](#)< _Iterator >::iterator_category [iterator_category](#)
- typedef _Iterator [iterator_type](#)
- typedef __traits_type::pointer [pointer](#)
- typedef __traits_type::reference [reference](#)
- typedef [iterator_traits](#)< _Iterator >::value_type [value_type](#)

Public Member Functions

- [reverse_iterator](#) ()
- [reverse_iterator](#) (iterator_type __x)
- template<typename _Iter >
 [reverse_iterator](#) (const [reverse_iterator](#)< _Iter > &__x)
- [reverse_iterator](#) (const [reverse_iterator](#) &__x)
- iterator_type [base](#) () const
- [reference](#) operator* () const
- [reverse_iterator](#) operator+ (difference_type __n) const
- [reverse_iterator](#) & operator++ ()
- [reverse_iterator](#) operator++ (int)
- [reverse_iterator](#) & operator+= (difference_type __n)
- [reverse_iterator](#) operator- (difference_type __n) const
- [reverse_iterator](#) & operator-- ()
- [reverse_iterator](#) operator-- (int)
- [reverse_iterator](#) & operator-= (difference_type __n)
- [pointer](#) operator-> () const
- [reference](#) operator[] (difference_type __n) const

Protected Types

- typedef [iterator_traits](#)< _Iterator > [__traits_type](#)

Protected Attributes

- `_Iterator current`

5.644.1 Detailed Description

template<typename `_Iterator`> class `std::reverse_iterator`< `_Iterator` >

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 95 of file `stl_iterator.h`.

5.644.2 Member Typedef Documentation

5.644.2.1 `template<typename _Iterator> typedef __traits_difference_type std::reverse_iterator< _Iterator >::difference_type`

Distance between iterators is represented as this type.

Reimplemented from `std::iterator`< `iterator_traits`< `_Iterator` >::`iterator_category`, `iterator_traits`< `_Iterator` >::`value_type`, `iterator_traits`< `_Iterator` >::`difference_type`, `iterator_traits`< `_Iterator` >::`pointer`, `iterator_traits`< `_Iterator` >::`reference` >.

Definition at line 109 of file `stl_iterator.h`.

5.644.2.2 `typedef iterator_traits< _Iterator >::iterator_category std::iterator< iterator_traits< _Iterator >::iterator_category , iterator_traits< _Iterator >::value_type , iterator_traits< _Iterator >::difference_type , iterator_traits< _Iterator >::pointer , iterator_traits< _Iterator >::reference >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 110 of file `stl_iterator_base_types.h`.

5.644.2.3 `template<typename _Iterator> typedef __traits_type::pointer
std::reverse_iterator<_Iterator>::pointer`

This type represents a pointer-to-`value_type`.

Reimplemented from `std::iterator< iterator_traits<_Iterator>::iterator_category,
iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type,
iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference >`.

Definition at line 110 of file `stl_iterator.h`.

5.644.2.4 `template<typename _Iterator> typedef __traits_type::reference
std::reverse_iterator<_Iterator>::reference`

This type represents a reference-to-`value_type`.

Reimplemented from `std::iterator< iterator_traits<_Iterator>::iterator_category,
iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type,
iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference >`.

Definition at line 111 of file `stl_iterator.h`.

5.644.2.5 `typedef iterator_traits<_Iterator>::value_type std::iterator<
iterator_traits<_Iterator>::iterator_category , iterator_traits<
_Iterator>::value_type , iterator_traits<_Iterator
>::difference_type , iterator_traits<_Iterator>::pointer
, iterator_traits<_Iterator>::reference >::value_type
[inherited]`

The type "pointed to" by the iterator.

Definition at line 112 of file `stl_iterator_base_types.h`.

5.644.3 Constructor & Destructor Documentation

5.644.3.1 `template<typename _Iterator> std::reverse_iterator<_Iterator
>::reverse_iterator () [inline]`

The default constructor default-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 119 of file `stl_iterator.h`.

5.644.3.2 `template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator (iterator_type __x) [inline, explicit]`

This iterator will move in the opposite direction that `x` does.

Definition at line 125 of file `stl_iterator.h`.

5.644.3.3 `template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator (const reverse_iterator< _Iterator > & __x) [inline]`

The copy constructor is normal.

Definition at line 130 of file `stl_iterator.h`.

5.644.3.4 `template<typename _Iterator> template<typename _Iter > std::reverse_iterator< _Iterator >::reverse_iterator (const reverse_iterator< _Iter > & __x) [inline]`

A [reverse_iterator](#) across other types can be copied in the normal fashion.

Definition at line 138 of file `stl_iterator.h`.

5.644.4 Member Function Documentation

5.644.4.1 `template<typename _Iterator> iterator_type std::reverse_iterator< _Iterator >::base () const [inline]`

Returns

`current`, the iterator used for underlying work.

Definition at line 145 of file `stl_iterator.h`.

Referenced by `std::operator==()`.

5.644.4.2 `template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator* () const [inline]`

Returns

TODO

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 154 of file stl_iterator.h.

```
5.644.4.3 template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator+ ( difference_type __n  
) const [inline]
```

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 225 of file stl_iterator.h.

```
5.644.4.4 template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator++ ( ) [inline]
```

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 175 of file stl_iterator.h.

```
5.644.4.5 template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator++ ( int ) [inline]
```

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 187 of file stl_iterator.h.

5.644.4.6 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator+=(difference_type
__n) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 234 of file stl_iterator.h.

5.644.4.7 `template<typename _Iterator> reverse_iterator
std::reverse_iterator< _Iterator >::operator-(difference_type __n
) const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 246 of file stl_iterator.h.

5.644.4.8 `template<typename _Iterator> reverse_iterator
std::reverse_iterator< _Iterator >::operator--(int) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 212 of file stl_iterator.h.

5.644.4.9 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator-- () [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 200 of file stl_iterator.h.

5.644.4.10 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator= (difference_type
__n) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 255 of file stl_iterator.h.

5.644.4.11 `template<typename _Iterator> pointer std::reverse_iterator<
_Iterator >::operator-> () const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 166 of file stl_iterator.h.

5.644.4.12 `template<typename _Iterator> reference std::reverse_iterator<
_Iterator >::operator[] (difference_type __n) const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 267 of file stl_iterator.h.

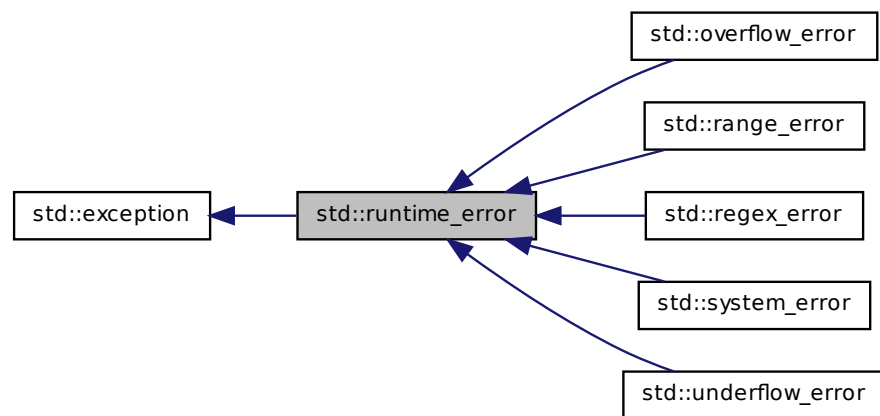
The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

5.645 std::runtime_error Class Reference

One of two subclasses of exception.

Inheritance diagram for std::runtime_error:



Public Member Functions

- `runtime_error` (const `string` &__arg)
- virtual const char * `what` () const throw ()

5.645.1 Detailed Description

One of two subclasses of exception. Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 107 of file `stdexcept`.

5.645.2 Constructor & Destructor Documentation

5.645.2.1 `std::runtime_error::runtime_error (const string & __arg)`
[**explicit**]

Takes a character string describing the error.

5.645.3 Member Function Documentation

5.645.3.1 `virtual const char* std::runtime_error::what () const throw ()`
[**virtual**]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.646 `std::seed_seq` Class Reference

The [seed_seq](#) class generates sequences of seeds for random number generators.

Public Types

- typedef uint_least32_t [result_type](#)

Public Member Functions

- [seed_seq](#) ()
- template<typename _IntType >
[seed_seq](#) ([std::initializer_list](#)< _IntType > il)

- `template<typename _InputIterator >`
`seed_seq` (`_InputIterator __begin`, `_InputIterator __end`)
- `template<typename _RandomAccessIterator >`
`void generate` (`_RandomAccessIterator __begin`, `_RandomAccessIterator __-`
`end`)
- `template<typename OutputIterator >`
`void param` (`OutputIterator __dest`) `const`
- `size_t size` () `const`

5.646.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 5300 of file `random.h`.

5.646.2 Member Typedef Documentation

5.646.2.1 `typedef uint_least32_t std::seed_seq::result_type`

The type of the seed vales.

Definition at line 5305 of file `random.h`.

5.646.3 Constructor & Destructor Documentation

5.646.3.1 `std::seed_seq::seed_seq () [inline]`

Default constructor.

Definition at line 5308 of file `random.h`.

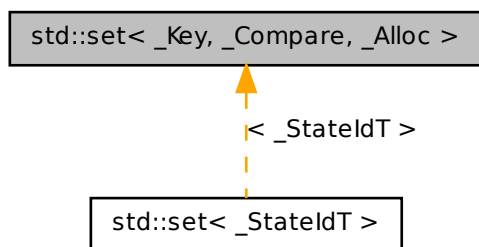
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.647 `std::set< _Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Inheritance diagram for std::set< _Key, _Compare, _Alloc >:



Public Types

- typedef `_Key` [key_type](#)
- typedef `_Key` [value_type](#)
- typedef `_Compare` [key_compare](#)
- typedef `_Compare` [value_compare](#)
- typedef `_Alloc` [allocator_type](#)
- typedef `_Key_alloc_type::pointer` [pointer](#)
- typedef `_Key_alloc_type::const_pointer` [const_pointer](#)
- typedef `_Key_alloc_type::reference` [reference](#)
- typedef `_Key_alloc_type::const_reference` [const_reference](#)
- typedef `_Rep_type::const_iterator` [iterator](#)
- typedef `_Rep_type::const_iterator` [const_iterator](#)
- typedef `_Rep_type::const_reverse_iterator` [reverse_iterator](#)
- typedef `_Rep_type::const_reverse_iterator` [const_reverse_iterator](#)
- typedef `_Rep_type::size_type` [size_type](#)
- typedef `_Rep_type::difference_type` [difference_type](#)

Public Member Functions

- [set](#) ()
- [set](#) (const `_Compare` &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- template<typename `_InputIterator` >
[set](#) (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- [set](#) (const [set](#) &__x)

- `template<typename _InputIterator >`
`set (_InputIterator __first, _InputIterator __last)`
- `set (set &&__x)`
- `set (initializer_list< value_type > __l, const _Compare &__comp=_Compare(),`
`const allocator_type &__a=allocator_type())`
- `iterator begin () const`
- `iterator cbegin () const`
- `iterator cend () const`
- `void clear ()`
- `size_type count (const key_type &__x) const`
- `reverse_iterator crbegin () const`
- `reverse_iterator crend () const`
- `bool empty () const`
- `iterator end () const`
- `iterator erase (iterator __first, iterator __last)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __position)`
- `allocator_type get_allocator () const`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `iterator insert (iterator __position, const value_type &__x)`
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `key_compare key_comp () const`
- `size_type max_size () const`
- `set & operator= (const set &__x)`
- `set & operator= (set &&__x)`
- `set & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () const`
- `reverse_iterator rend () const`
- `size_type size () const`
- `void swap (set &__x)`
- `value_compare value_comp () const`

- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`

- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`

- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__-`
`x) const`

Friends

- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 >`
`&)`
- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 >`
`&)`

5.647.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set< _Key, _Compare, _Alloc >`

A standard container made up of unique keys, which can be retrieved in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

Parameters

Key Type of key objects.

Compare Comparison function object type, defaults to `less<Key>`.

Alloc Allocator type, defaults to `allocator<Key>`.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 87 of file `stl_set.h`.

5.647.2 Member Typedef Documentation

5.647.2.1 `template<typename _Key, typename _Compare = std::less<_Key>,`
`typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set<`
`_Key, _Compare, _Alloc >::allocator_type`

Public typedefs.

Definition at line 104 of file `stl_set.h`.

5.647.2.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc
>::const_iterator`

Iterator-related typedefs.

Definition at line 125 of file stl_set.h.

5.647.2.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Key_alloc_type::const_pointer std::set< _Key, _Compare, _Alloc
>::const_pointer`

Iterator-related typedefs.

Definition at line 118 of file stl_set.h.

5.647.2.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Key_alloc_type::const_reference std::set< _Key, _Compare, _Alloc
>::const_reference`

Iterator-related typedefs.

Definition at line 120 of file stl_set.h.

5.647.2.5 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc
>::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 127 of file stl_set.h.

5.647.2.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::difference_type std::set< _Key, _Compare, _Alloc
>::difference_type`

Iterator-related typedefs.

Definition at line 129 of file stl_set.h.

5.647.2.7 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> typedef
 _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc
 >::iterator`

Iterator-related typedefs.

Definition at line 124 of file stl_set.h.

5.647.2.8 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> typedef _Compare
 std::set< _Key, _Compare, _Alloc >::key_compare`

Public typedefs.

Definition at line 102 of file stl_set.h.

5.647.2.9 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> typedef _Key std::set<
 _Key, _Compare, _Alloc >::key_type`

Public typedefs.

Definition at line 100 of file stl_set.h.

5.647.2.10 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> typedef
 _Key_alloc_type::pointer std::set< _Key, _Compare, _Alloc
 >::pointer`

Iterator-related typedefs.

Definition at line 117 of file stl_set.h.

5.647.2.11 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> typedef
 _Key_alloc_type::reference std::set< _Key, _Compare, _Alloc
 >::reference`

Iterator-related typedefs.

Definition at line 119 of file stl_set.h.

5.647.2.12 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_reverse_iterator std::set< _Key, _Compare,
_Alloc >::reverse_iterator`

Iterator-related typedefs.

Definition at line 126 of file stl_set.h.

5.647.2.13 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::size_type std::set< _Key, _Compare, _Alloc
>::size_type`

Iterator-related typedefs.

Definition at line 128 of file stl_set.h.

5.647.2.14 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef _Compare
std::set< _Key, _Compare, _Alloc >::value_compare`

Public typedefs.

Definition at line 103 of file stl_set.h.

5.647.2.15 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<
_Key, _Compare, _Alloc >::value_type`

Public typedefs.

Definition at line 101 of file stl_set.h.

5.647.3 Constructor & Destructor Documentation

5.647.3.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::set< _Key,
_Compare, _Alloc >::set () [inline]`

Default constructor creates no elements.

Definition at line 136 of file stl_set.h.

```

5.647.3.2  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set< _Key,
              _Compare, _Alloc >::set ( const _Compare & __comp, const
              allocator_type & __a = allocator_type () ) [inline,
              explicit]

```

Creates a set with no elements.

Parameters

comp Comparator to use.

a An allocator object.

Definition at line 145 of file stl_set.h.

```

5.647.3.3  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> template<typename
              _InputIterator > std::set< _Key, _Compare, _Alloc >::set (
              _InputIterator __first, _InputIterator __last ) [inline]

```

Builds a set from a range.

Parameters

first An input iterator.

last An input iterator.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 159 of file stl_set.h.

```

5.647.3.4  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> template<typename
              _InputIterator > std::set< _Key, _Compare, _Alloc >::set (
              _InputIterator __first, _InputIterator __last, const _Compare
              & __comp, const allocator_type & __a = allocator_type () )
              [inline]

```

Builds a set from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 175 of file stl_set.h.

```
5.647.3.5  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set<_Key,
              _Compare, _Alloc >::set ( const set<_Key, _Compare, _Alloc > &
              __x ) [inline]
```

Set copy constructor.

Parameters

x A set of identical element and allocator types.

The newly-created set uses a copy of the allocation object used by *x*.

Definition at line 188 of file stl_set.h.

```
5.647.3.6  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set<_Key,
              _Compare, _Alloc >::set ( set<_Key, _Compare, _Alloc > && __x
              ) [inline]
```

Set move constructor

Parameters

x A set of identical element and allocator types.

The newly-created set contains the exact contents of *x*. The contents of *x* are a valid, but unspecified set.

Definition at line 199 of file stl_set.h.

```
5.647.3.7  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set<_Key,
              _Compare, _Alloc >::set ( initializer_list<value_type> & __l, const
              _Compare & __comp = _Compare(), const allocator_type & __a =
              allocator_type() ) [inline]
```

Builds a set from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison functor.

a An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 212 of file stl_set.h.

5.647.4 Member Function Documentation

5.647.4.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 292 of file stl_set.h.

5.647.4.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 329 of file stl_set.h.

5.647.4.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 338 of file stl_set.h.

5.647.4.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::set<_Key,
_Compare, _Alloc>::clear () [inline]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 552 of file `stl_set.h`.

5.647.4.5 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::set<
_Key, _Compare, _Alloc>::count (const key_type & __x) const
[inline]`

Finds the number of elements.

Parameters

x Element to located.

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 566 of file `stl_set.h`.

5.647.4.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<
_Key, _Compare, _Alloc>::crbegin () const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 347 of file `stl_set.h`.

5.647.4.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<
_Key, _Compare, _Alloc>::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 356 of file `stl_set.h`.

5.647.4.8 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> bool std::set< _Key,
_Compare, _Alloc >::empty () const [inline]`

Returns true if the set is empty.

Definition at line 362 of file stl_set.h.

5.647.4.9 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
_Compare, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 301 of file stl_set.h.

5.647.4.10 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::pair<iterator,
iterator> std::set< _Key, _Compare, _Alloc >::equal_range (const
key_type & __x) [inline]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 646 of file stl_set.h.

5.647.4.11 `template<typename _Key, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::set<_Key,
_Compare, _Alloc>::equal_range (const key_type & __x) const
[inline]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 650 of file `stl_set.h`.

5.647.4.12 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::erase (iterator __position) [inline]`

Erases an element from a set.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 478 of file `stl_set.h`.

5.647.4.13 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::set< _Key,
_Compare, _Alloc >::erase (const key_type & __x) [inline]`

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 507 of file stl_set.h.

5.647.4.14 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
_Compare, _Alloc >::erase (iterator __first, iterator __last)
[inline]`

Erases a [first,last) range of elements from a set.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 526 of file stl_set.h.

5.647.4.15 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
_Compare, _Alloc >::find (const key_type & __x) [inline]`

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 584 of file `stl_set.h`.

```
5.647.4.16  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> const_iterator std::set<
              _Key, _Compare, _Alloc >::find ( const key_type & __x ) const
              [inline]
```

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 588 of file `stl_set.h`.

```
5.647.4.17  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> allocator_type std::set<
              _Key, _Compare, _Alloc >::get_allocator ( ) const [inline]
```

Returns the allocator object with which the set was constructed.

Definition at line 283 of file `stl_set.h`.

```
5.647.4.18  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::pair<iterator,
              bool> std::set< _Key, _Compare, _Alloc >::insert ( const
              value_type & __x ) [inline]
```

Attempts to insert an element into the set.

Parameters

x Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 405 of file stl_set.h.

```
5.647.4.19 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
_Compare, _Alloc >::insert ( iterator __position, const value_type
& __x ) [inline]
```

Attempts to insert an element into the set.

Parameters

position An iterator that serves as a hint as to where the element should be inserted.

x Element to be inserted.

Returns

An iterator that points to the element with key of *x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 432 of file stl_set.h.

5.647.4.20 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>>> template<typename
_InputIterator > void std::set< _Key, _Compare, _Alloc >::insert (
_InputIterator __first, _InputIterator __last) [inline]`

A template function that attempts to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 446 of file stl_set.h.

5.647.4.21 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>>> void std::set< _Key,
_Compare, _Alloc >::insert (initializer_list< value_type > __l)
[inline]`

Attempts to insert a list of elements into the set.

Parameters

list A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 458 of file stl_set.h.

Referenced by std::set< _StateIdT >::insert().

5.647.4.22 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>>> key_compare std::set<
_Key, _Compare, _Alloc >::key_comp () const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 275 of file stl_set.h.

5.647.4.23 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>>> iterator std::set< _Key,
_Compare, _Alloc >::lower_bound (const key_type & __x)
[inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 605 of file stl_set.h.

```
5.647.4.24  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>>> const_iterator std::set<
              _Key, _Compare, _Alloc >::lower_bound ( const key_type & __x )
              const [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 609 of file stl_set.h.

```
5.647.4.25  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>>> size_type std::set<
              _Key, _Compare, _Alloc >::max_size ( ) const [inline]
```

Returns the maximum size of the set.

Definition at line 372 of file stl_set.h.

5.647.4.26 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,
_Compare, _Alloc >::operator= (set< _Key, _Compare, _Alloc >
&& __x) [inline]`

Set move assignment operator.

Parameters

x A set of identical element and allocator types.

The contents of *x* are moved into this set (without copying). *x* is a valid, but unspecified set.

Definition at line 242 of file `stl_set.h`.

5.647.4.27 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,
_Compare, _Alloc >::operator= (initializer_list< value_type > __l
) [inline]`

Set list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a set with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 263 of file `stl_set.h`.

5.647.4.28 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,
_Compare, _Alloc >::operator= (const set< _Key, _Compare,
_Alloc > & __x) [inline]`

Set assignment operator.

Parameters

x A set of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 227 of file `stl_set.h`.

5.647.4.29 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::set< _Key, _Compare, _Alloc >::rbegin () const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 310 of file stl_set.h.

5.647.4.30 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::set< _Key, _Compare, _Alloc >::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 319 of file stl_set.h.

5.647.4.31 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::set<
_Key, _Compare, _Alloc >::size () const [inline]`

Returns the size of the set.

Definition at line 367 of file stl_set.h.

5.647.4.32 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::set< _Key,
_Compare, _Alloc >::swap (set< _Key, _Compare, _Alloc > & __x
) [inline]`

Swaps data with another set.

Parameters

x A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 387 of file stl_set.h.

Referenced by `std::swap()`.

5.647.4.33 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc >::upper_bound (const key_type & __x)
[inline]`

Finds the end of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 621 of file `stl_set.h`.

5.647.4.34 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator std::set<
_Key, _Compare, _Alloc >::upper_bound (const key_type & __x)
const [inline]`

Finds the end of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 625 of file `stl_set.h`.

5.647.4.35 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> value_compare std::set<
_Key, _Compare, _Alloc >::value_comp () const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 279 of file `stl_set.h`.

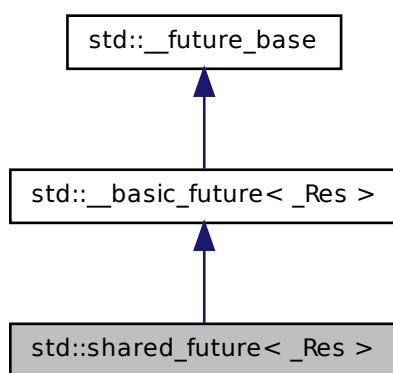
The documentation for this class was generated from the following file:

- [stl_set.h](#)

5.648 `std::shared_future< _Res >` Class Template Reference

Primary template for `shared_future`.

Inheritance diagram for `std::shared_future< _Res >`:



Public Member Functions

- `shared_future` (const `shared_future` &__sf)
- `shared_future` (`shared_future` &&__sf)
- `shared_future` (`future`< _Res > &&__uf)
- const _Res & `get` ()
- `shared_future` & `operator=` (const `shared_future` &__sf)
- `shared_future` & `operator=` (`shared_future` &&__sf)
- bool `valid` () const
- void `wait` () const
- template<typename _Rep , typename _Period >
bool `wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- template<typename _Clock , typename _Duration >
bool `wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs)
const

Protected Types

- typedef [__future_base::Result](#)< [_Res](#) > & [__result_type](#)
- typedef [shared_ptr](#)< [_State](#) > [__state_type](#)

Protected Member Functions

- [__result_type](#) [_M_get_result](#) ()
- void [_M_swap](#) ([__basic_future](#) &__that)

5.648.1 Detailed Description

template<typename [_Res](#)> class [std::shared_future](#)< [_Res](#) >

Primary template for [shared_future](#).

Definition at line 683 of file future.

5.648.2 Constructor & Destructor Documentation

5.648.2.1 template<typename [_Res](#) > [std::shared_future](#)< [_Res](#) >::[shared_future](#) (const [shared_future](#)< [_Res](#) > & [__sf](#))
[inline]

Copy constructor.

Definition at line 691 of file future.

5.648.2.2 template<typename [_Res](#) > [std::shared_future](#)< [_Res](#) >::[shared_future](#) ([future](#)< [_Res](#) > && [__uf](#)) [inline]

Construct from a future rvalue.

Definition at line 694 of file future.

5.648.2.3 template<typename [_Res](#) > [std::shared_future](#)< [_Res](#) >::[shared_future](#) ([shared_future](#)< [_Res](#) > && [__sf](#)) [inline]

Construct from a [shared_future](#) rvalue.

Definition at line 699 of file future.

5.648.3 Member Function Documentation

5.648.3.1 `template<typename _Res> __result_type std::__basic_future< _Res >::_M_get_result() [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

5.648.3.2 `template<typename _Res > const _Res& std::shared_future< _Res >::get() [inline]`

Retrieving the value.

Definition at line 717 of file future.

References `std::__basic_future< _Res >::_M_get_result()`.

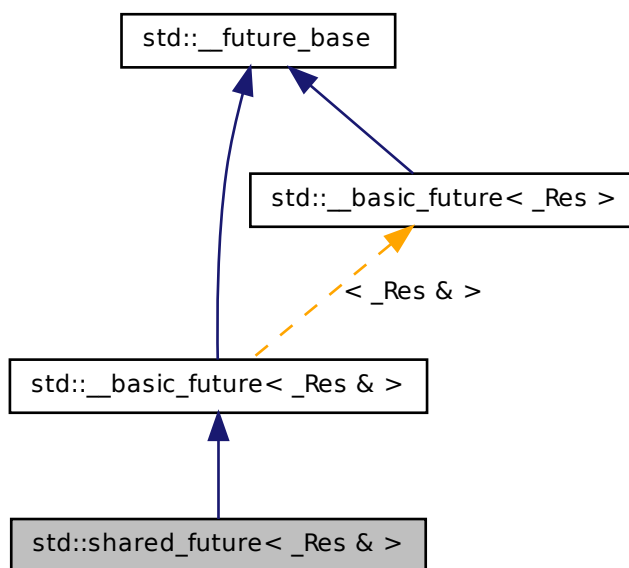
The documentation for this class was generated from the following file:

- [future](#)

5.649 `std::shared_future< _Res & >` Class Template Reference

Partial specialization for `shared_future<R&>`

Inheritance diagram for `std::shared_future< _Res & >`:



Public Member Functions

- `shared_future` (const `shared_future` &__sf)
- `shared_future` (`shared_future` &&__sf)
- `shared_future` (`future`< _Res & > &&__uf)
- `_Res` & `get` ()
- `shared_future` & `operator=` (const `shared_future` &__sf)
- `shared_future` & `operator=` (`shared_future` &&__sf)
- `bool valid` () const
- `void wait` () const
- `bool wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- `bool wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs) const

Protected Types

- typedef [__future_base::_Result](#)< _Res & > & [__result_type](#)
- typedef [shared_ptr](#)< [_State](#) > [__state_type](#)

Protected Member Functions

- [__result_type](#) [_M_get_result](#) ()
- void [_M_swap](#) ([__basic_future](#) &__that)

5.649.1 Detailed Description

template<typename _Res> class std::shared_future< _Res & >

Partial specialization for shared_future<R&>

Definition at line 727 of file future.

5.649.2 Constructor & Destructor Documentation

5.649.2.1 **template<typename _Res > std::shared_future< _Res & >::shared_future (const shared_future< _Res & > & __sf)**
[inline]

Copy constructor.

Definition at line 735 of file future.

5.649.2.2 **template<typename _Res > std::shared_future< _Res & >::shared_future (future< _Res & > && __uf)** **[inline]**

Construct from a future rvalue.

Definition at line 738 of file future.

5.649.2.3 **template<typename _Res > std::shared_future< _Res & >::shared_future (shared_future< _Res & > && __sf)**
[inline]

Construct from a [shared_future](#) rvalue.

Definition at line 743 of file future.

5.649.3 Member Function Documentation

5.649.3.1 `__result_type std::__basic_future<_Res & >::_M_get_result ()` `[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

References `std::rethrow_exception()`.

5.649.3.2 `template<typename _Res > _Res& std::shared_future<_Res & >::get () [inline]`

Retrieving the value.

Definition at line 761 of file future.

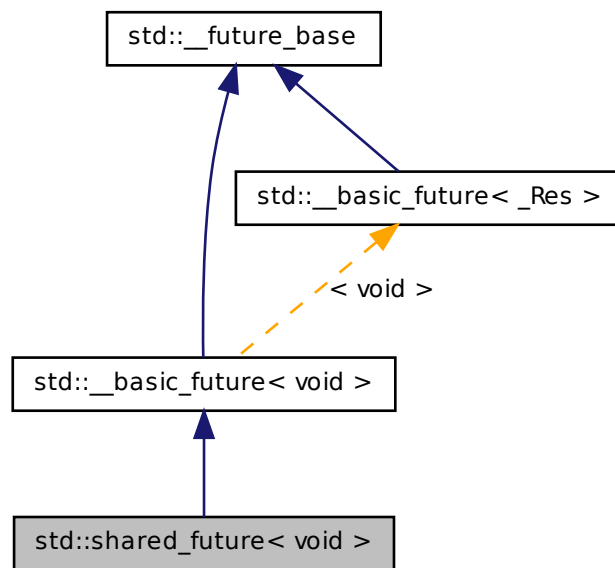
The documentation for this class was generated from the following file:

- [future](#)

5.650 `std::shared_future< void >` Class Template Reference

Explicit specialization for [shared_future<void>](#)

Inheritance diagram for `std::shared_future< void >`:



Public Member Functions

- `shared_future` (const `shared_future` &__sf)
- `shared_future` (`shared_future` &&__sf)
- `shared_future` (`future`< void > &&__uf)
- void `get` ()
- `shared_future` & `operator=` (const `shared_future` &__sf)
- `shared_future` & `operator=` (`shared_future` &&__sf)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- bool `wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs) const

Protected Types

- typedef [__future_base::_Result](#)< void > & [__result_type](#)
- typedef [shared_ptr](#)< [_State](#) > [__state_type](#)

Protected Member Functions

- [__result_type](#) [_M_get_result](#) ()
- void [_M_swap](#) ([__basic_future](#) &__that)

5.650.1 Detailed Description

`template<> class std::shared_future< void >`

Explicit specialization for [shared_future](#)<void>

Definition at line 766 of file future.

5.650.2 Constructor & Destructor Documentation

5.650.2.1 `std::shared_future< void >::shared_future (const shared_future< void > & __sf) [inline]`

Copy constructor.

Definition at line 774 of file future.

5.650.2.2 `std::shared_future< void >::shared_future (future< void > && __uf) [inline]`

Construct from a future rvalue.

Definition at line 777 of file future.

5.650.2.3 `std::shared_future< void >::shared_future (shared_future< void > && __sf) [inline]`

Construct from a [shared_future](#) rvalue.

Definition at line 782 of file future.

5.650.3 Member Function Documentation

5.650.3.1 `__result_type std::__basic_future< void >::__M_get_result ()`
[inline, protected, inherited]

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

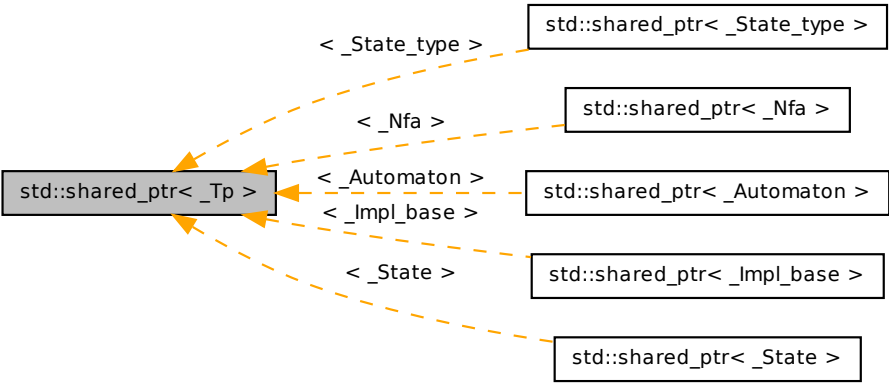
The documentation for this class was generated from the following file:

- [future](#)

5.651 `std::shared_ptr<_Tp>` Class Template Reference

A smart pointer with reference-counted copy semantics.

Inheritance diagram for `std::shared_ptr<_Tp>`:



Public Member Functions

- [shared_ptr\(\)](#)
- `template<typename _Tp1>`
[shared_ptr](#) (`_Tp1 *__p`)

- `template<typename _Deleter >`
`shared_ptr` (`nullptr_t __p`, `_Deleter __d`)
- `template<typename _Tp1 >`
`shared_ptr` (`const shared_ptr<_Tp1 > &__r`)
- `shared_ptr` (`shared_ptr &&__r`)
- `template<typename _Tp1 , typename _Deleter , typename _Alloc >`
`shared_ptr` (`_Tp1 *__p`, `_Deleter __d`, `const _Alloc &__a`)
- `shared_ptr` (`nullptr_t __p`)
- `template<typename _Tp1 , typename _Del >`
`shared_ptr` (`std::unique_ptr<_Tp1, _Del > &&__r`)
- `template<typename _Tp1 >`
`shared_ptr` (`std::auto_ptr<_Tp1 > &&__r`)
- `template<typename _Tp1 >`
`shared_ptr` (`shared_ptr<_Tp1 > &&__r`)
- `template<typename _Tp1 >`
`shared_ptr` (`const weak_ptr<_Tp1 > &__r`)
- `template<typename _Tp1 , typename _Deleter >`
`shared_ptr` (`_Tp1 *__p`, `_Deleter __d`)
- `template<typename _Deleter , typename _Alloc >`
`shared_ptr` (`nullptr_t __p`, `_Deleter __d`, `const _Alloc &__a`)
- `template<typename _Tp1 >`
`shared_ptr` (`const shared_ptr<_Tp1 > &__r`, `_Tp *__p`)
- `template<class _Tp1 >`
`shared_ptr & operator=` (`shared_ptr<_Tp1 > &&__r`)
- `template<typename _Tp1 >`
`shared_ptr & operator=` (`const shared_ptr<_Tp1 > &__r`)
- `shared_ptr & operator=` (`shared_ptr &&__r`)
- `template<typename _Tp1 , typename _Del >`
`shared_ptr & operator=` (`std::unique_ptr<_Tp1, _Del > &&__r`)
- `template<typename _Tp1 >`
`shared_ptr & operator=` (`std::auto_ptr<_Tp1 > &&__r`)

Friends

- `template<typename _Tp1 , typename _Alloc , typename... _Args >`
`shared_ptr<_Tp1 > allocate_shared` (`_Alloc __a`, `_Args &&...__args`)

5.651.1 Detailed Description

`template<typename _Tp> class std::shared_ptr<_Tp >`

A smart pointer with reference-counted copy semantics. The object pointed to is deleted when the last `shared_ptr` pointing to it is destroyed or reset.

Definition at line 91 of file `shared_ptr.h`.

5.651.2 Constructor & Destructor Documentation

5.651.2.1 `template<typename _Tp> std::shared_ptr< _Tp >::shared_ptr ()
[inline]`

Construct an empty shared_ptr.

Postcondition

`use_count()==0 && get()==0`

Definition at line 98 of file shared_ptr.h.

5.651.2.2 `template<typename _Tp> template<typename _Tp1 >
std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p) [inline,
explicit]`

Construct a shared_ptr that owns the pointer `__p`.

Parameters

`__p` A pointer that is convertible to `element_type*`.

Postcondition

`use_count() == 1 && get() == __p`

Exceptions

`std::bad_alloc`, in which case `delete __p` is called.

Definition at line 107 of file shared_ptr.h.

5.651.2.3 `template<typename _Tp> template<typename _Tp1 , typename
_Deleter > std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p,
_Deleter __d) [inline]`

Construct a shared_ptr that owns the pointer `__p` and the deleter `__d`.

Parameters

`__p` A pointer.

`__d` A deleter.

Postcondition

`use_count() == 1 && get() == __p`

Exceptions

std::bad_alloc, in which case `__d(__p)` is called.

Requirements: `_Deleter`'s copy constructor and destructor must not throw

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 123 of file `shared_ptr.h`.

5.651.2.4 `template<typename _Tp> template<typename _Deleter >`
`std::shared_ptr< _Tp >::shared_ptr (nullptr_t __p, _Deleter __d`
`) [inline]`

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

Parameters

`__p` A null pointer constant.

`__d` A deleter.

Postcondition

`use_count() == 1 && get() == __p`

Exceptions

std::bad_alloc, in which case `__d(__p)` is called.

Requirements: `_Deleter`'s copy constructor and destructor must not throw

The last owner will call `__d(__p)`

Definition at line 139 of file `shared_ptr.h`.

5.651.2.5 `template<typename _Tp> template<typename _Tp1, typename`
`_Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr (`
`_Tp1 * __p, _Deleter __d, const _Alloc & __a) [inline]`

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

Parameters

`__p` A pointer.

`__d` A deleter.

`__a` An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc, in which case __d(__p) is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

__shared_ptr will release __p by calling __d(__p)

Definition at line 158 of file shared_ptr.h.

5.651.2.6 `template<typename _Tp> template<typename _Deleter , typename
_Alloc > std::shared_ptr< _Tp >::shared_ptr (nullptr_t __p,
_Deleter __d, const _Alloc & __a) [inline]`

Construct a shared_ptr that owns a null pointer and the deleter __d.

Parameters

__p A null pointer constant.

__d A deleter.

__a An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc, in which case __d(__p) is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

The last owner will call __d(__p)

Definition at line 177 of file shared_ptr.h.

5.651.2.7 `template<typename _Tp> template<typename _Tp1 >
std::shared_ptr< _Tp >::shared_ptr (const shared_ptr< _Tp1 > &
__r, _Tp * __p) [inline]`

Constructs a shared_ptr instance that stores __p and shares ownership with __r.

Parameters

`__r` A `shared_ptr`.

`__p` A pointer that will remain valid while `*__r` is valid.

Postcondition

`get() == __p && use_count() == __r.use_count()`

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 199 of file `shared_ptr.h`.

5.651.2.8 `template<typename _Tp> template<typename _Tp1 >`
`std::shared_ptr<_Tp>::shared_ptr (const shared_ptr<_Tp1> &`
`__r) [inline]`

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

Parameters

`__r` A `shared_ptr`.

Postcondition

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 210 of file `shared_ptr.h`.

5.651.2.9 `template<typename _Tp> std::shared_ptr<_Tp>::shared_ptr (`
`shared_ptr<_Tp> && __r) [inline]`

Move-constructs a `shared_ptr` instance from `__r`.

Parameters

`__r` A `shared_ptr` rvalue.

Postcondition

`*this` contains the old value of `__r`, `__r` is empty.

Definition at line 217 of file `shared_ptr.h`.

5.651.2.10 `template<typename _Tp> template<typename _Tp1 >
std::shared_ptr< _Tp >::shared_ptr (shared_ptr< _Tp1 > &&
__r) [inline]`

Move-constructs a shared_ptr instance from __r.

Parameters

`__r` A shared_ptr rvalue.

Postcondition

*this contains the old value of __r, __r is empty.

Definition at line 226 of file shared_ptr.h.

5.651.2.11 `template<typename _Tp> template<typename _Tp1 >
std::shared_ptr< _Tp >::shared_ptr (const weak_ptr< _Tp1 > &
__r) [inline, explicit]`

Constructs a shared_ptr that shares ownership with __r and stores a copy of the pointer stored in __r.

Parameters

`__r` A [weak_ptr](#).

Postcondition

`use_count() == __r.use_count()`

Exceptions

bad_weak_ptr when __r.expired(), in which case the constructor has no effect.

Definition at line 238 of file shared_ptr.h.

5.651.2.12 `template<typename _Tp> std::shared_ptr< _Tp >::shared_ptr (
nullptr_t __p) [inline]`

Construct an empty shared_ptr.

Parameters

`__p` A null pointer constant.

Postcondition

`use_count() == 0 && get() == nullptr`

Definition at line 256 of file `shared_ptr.h`.

5.651.3 Friends And Related Function Documentation

5.651.3.1 `template<typename _Tp> template<typename _Tp1 , typename
_Alloc , typename... _Args> shared_ptr<_Tp1> allocate_shared (
_Alloc __a, _Args &&... __args) [friend]`

Create an object that is owned by a [shared_ptr](#).

Parameters

`__a` An allocator.

`__args` Arguments for the `_Tp` object's constructor.

Returns

A [shared_ptr](#) that owns the newly created object.

Exceptions

An exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the [shared_ptr](#) and the new object.

Definition at line 495 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

5.652 `std::shuffle_order_engine<_RandomNumberEngine, __k >` Class Template Reference

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Public Types

- `typedef _RandomNumberEngine::result_type` [result_type](#)

Public Member Functions

- `shuffle_order_engine` ()
- `shuffle_order_engine` (const `_RandomNumberEngine` &__rne)
- `shuffle_order_engine` (result_type __s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`
`shuffle_order_engine` (_Sseq &__q)
- `shuffle_order_engine` (_RandomNumberEngine &&__rne)
- const `_RandomNumberEngine` & `base` () const
- void `discard` (unsigned long long __z)
- result_type `max` () const
- result_type `min` () const
- result_type `operator()` ()
- void `seed` (result_type __s)
- `template<typename _Sseq >`
void `seed` (_Sseq &__q)
- void `seed` ()

Static Public Attributes

- static const size_t `table_size`

Friends

- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &)`
- bool `operator==` (const `shuffle_order_engine` &__lhs, const `shuffle_order_engine` &__rhs)
- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &)`

5.652.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __k> class std::shuffle_order_engine< _RandomNumberEngine, __k >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1234 of file random.h.

5.652.2 Member Typedef Documentation

5.652.2.1 `template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type`

The type of the generated random value.

Definition at line 1241 of file random.h.

5.652.3 Constructor & Destructor Documentation

5.652.3.1 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine () [inline]`

Constructs a default shuffle_order_engine engine.

The underlying engine is default constructed as well.

Definition at line 1250 of file random.h.

5.652.3.2 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (const _RandomNumberEngine & __rne) [inline, explicit]`

Copy constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

rng An existing (base class) engine object.

Definition at line 1261 of file random.h.

5.652.3.3 `template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine (_RandomNumberEngine && __rne)
[inline, explicit]`

Move constructs a `shuffle_order_engine` engine.

Copies an existing base class random number generator.

Parameters

rne An existing (base class) engine object.

Definition at line 1272 of file `random.h`.

5.652.3.4 `template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine (result_type __s) [inline,
explicit]`

Seed constructs a `shuffle_order_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

`__s` A seed value for the base class engine.

Definition at line 1283 of file `random.h`.

5.652.3.5 `template<typename _RandomNumberEngine, size_t
__k> template<typename _Sseq, typename = typename
std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value
&& !std::is_same<_Sseq, _RandomNumberEngine>::value>
::type> std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine (_Sseq & __q) [inline, explicit]`

Generator construct a `shuffle_order_engine` engine.

Parameters

`__q` A seed sequence.

Definition at line 1297 of file `random.h`.

5.652.4 Member Function Documentation

5.652.4.1 `template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine& std::shuffle_order_engine<
_RandomNumberEngine, __k >::base () const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 1340 of file random.h.

5.652.4.2 `template<typename _RandomNumberEngine, size_t __k> void
std::shuffle_order_engine< _RandomNumberEngine, __k >::discard
(unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 1367 of file random.h.

5.652.4.3 `template<typename _RandomNumberEngine, size_t __k>
result_type std::shuffle_order_engine< _RandomNumberEngine, __k
>::max () const [inline]`

Gets the maximum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1358 of file random.h.

5.652.4.4 `template<typename _RandomNumberEngine, size_t __k>
result_type std::shuffle_order_engine< _RandomNumberEngine, __k
>::min () const [inline]`

Gets the minimum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1349 of file random.h.

5.652.4.5 `template<typename _RandomNumberEngine, size_t __k>
shuffle_order_engine<_RandomNumberEngine, __k>::result_type
std::shuffle_order_engine<_RandomNumberEngine, __k
>::operator() ()`

Gets the next value in the generated random number sequence.

Definition at line 767 of file random.tcc.

5.652.4.6 `template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq > void std::shuffle_order_engine<
_RandomNumberEngine, __k>::seed (_Sseq & __q) [inline]`

Reseeds the shuffle_order_engine object with the given seed sequence.

Parameters

`__q` A seed generator function.

Definition at line 1330 of file random.h.

5.652.4.7 `template<typename _RandomNumberEngine, size_t __k> void
std::shuffle_order_engine<_RandomNumberEngine, __k>::seed (
result_type __s) [inline]`

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

Definition at line 1317 of file random.h.

5.652.4.8 `template<typename _RandomNumberEngine, size_t __k> void
std::shuffle_order_engine<_RandomNumberEngine, __k>::seed (
) [inline]`

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

Definition at line 1306 of file random.h.

5.652.5 Friends And Related Function Documentation

5.652.5.1 `template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1,
typename _CharT, typename _Traits > std::basic_ostream<_CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & ,
const std::shuffle_order_engine< _RandomNumberEngine1, __k1 >
&) [friend]`

Inserts the current state of a `shuffle_order_engine` random number generator engine `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `shuffle_order_engine` random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.652.5.2 `template<typename _RandomNumberEngine, size_t __k> bool
operator==(const shuffle_order_engine< _RandomNumberEngine,
__k > & __lhs, const shuffle_order_engine<
_RandomNumberEngine, __k > & __rhs) [friend]`

Compares two `shuffle_order_engine` random number generator objects of the same type for equality.

Parameters

`__lhs` A `shuffle_order_engine` random number generator object.

`__rhs` Another `shuffle_order_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1391 of file `random.h`.

```

5.652.5.3  template<typename _RandomNumberEngine, size_t __k>
            template<typename _RandomNumberEngine1, size_t __k1,
                    typename _CharT, typename _Traits > std::basic_istream<_CharT,
                    _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> &,
                    std::shuffle_order_engine<_RandomNumberEngine1, __k1> & )
            [friend]

```

Extracts the current state of a % subtract_with_carry_engine random number generator engine `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A shuffle_order_engine random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.653 std::slice Class Reference

Class defining one-dimensional subset of an array.

Public Member Functions

- [slice](#) ()
- [slice](#) (size_t, size_t, size_t)
- size_t [size](#) () const
- size_t [start](#) () const
- size_t [stride](#) () const

5.653.1 Detailed Description

Class defining one-dimensional subset of an array. The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 58 of file slice_array.h.

The documentation for this class was generated from the following file:

- [slice_array.h](#)

5.654 std::slice_array< _Tp > Class Template Reference

Reference to one-dimensional subset of an array.

Public Types

- typedef `_Tp` **value_type**

Public Member Functions

- [slice_array](#) (const [slice_array](#) &)
- template<class `_Dom` >
void **operator%=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator%=](#) (const [valarray](#)< `_Tp` > &) const
- void [operator&=](#) (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >
void **operator&=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- template<class `_Dom` >
void **operator*=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator*=](#) (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >
void **operator+=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator+=](#) (const [valarray](#)< `_Tp` > &) const
- void [operator-=](#) (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >
void **operator-=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- template<class `_Dom` >
void **operator/=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator/=](#) (const [valarray](#)< `_Tp` > &) const
- void [operator<<=](#) (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >
void **operator<<=** (const `_Expr`< `_Dom`, `_Tp` > &) const

- void `operator=` (const _Tp &) const
- `slice_array` & `operator=` (const `slice_array` &)
- template<class _Dom >
void `operator=` (const _Expr< _Dom, _Tp > &) const
- void `operator=` (const `valarray`< _Tp > &) const
- template<class _Dom >
void `operator>>=` (const _Expr< _Dom, _Tp > &) const
- void `operator>>=` (const `valarray`< _Tp > &) const
- template<class _Dom >
void `operator^=` (const _Expr< _Dom, _Tp > &) const
- void `operator^=` (const `valarray`< _Tp > &) const
- void `operator|=` (const `valarray`< _Tp > &) const
- template<class _Dom >
void `operator|=` (const _Expr< _Dom, _Tp > &) const

Friends

- class `valarray`< _Tp >

5.654.1 Detailed Description

`template<typename _Tp> class std::slice_array< _Tp >`

Reference to one-dimensional subset of an array. A `slice_array` is a reference to the actual elements of an array specified by a slice. The way to get a `slice_array` is to call `operator[]`(slice) on a `valarray`. The returned `slice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `slice_array` refers to.

Parameters

Tp Element type.

Definition at line 122 of file `slice_array.h`.

5.654.2 Member Function Documentation

5.654.2.1 `template<typename _Tp> void std::slice_array< _Tp >::operator%=(const valarray< _Tp > &) const`

Modulo slice elements by corresponding elements of *v*.

5.654.2.2 `template<typename _Tp> void std::slice_array< _Tp
>::operator&= (const valarray< _Tp > &) const`

Logical and slice elements with corresponding elements of *v*.

5.654.2.3 `template<typename _Tp> void std::slice_array< _Tp >::operator*=
(const valarray< _Tp > &) const`

Multiply slice elements by corresponding elements of *v*.

5.654.2.4 `template<typename _Tp> void std::slice_array< _Tp >::operator+=
(const valarray< _Tp > &) const`

Add corresponding elements of *v* to slice elements.

5.654.2.5 `template<typename _Tp> void std::slice_array< _Tp >::operator-=
(const valarray< _Tp > &) const`

Subtract corresponding elements of *v* from slice elements.

5.654.2.6 `template<typename _Tp> void std::slice_array< _Tp >::operator/=
(const valarray< _Tp > &) const`

Divide slice elements by corresponding elements of *v*.

5.654.2.7 `template<typename _Tp> void std::slice_array< _Tp
>::operator<<= (const valarray< _Tp > &) const`

Left shift slice elements by corresponding elements of *v*.

5.654.2.8 `template<typename _Tp> void std::slice_array< _Tp
>::operator>>= (const valarray< _Tp > &) const`

Right shift slice elements by corresponding elements of *v*.

5.654.2.9 `template<typename _Tp> void std::slice_array< _Tp >::operator^=
(const valarray< _Tp > &) const`

Logical xor slice elements with corresponding elements of *v*.

5.654.2.10 `template<typename _Tp> void std::slice_array< _Tp
>::operator|= (const valarray< _Tp > &) const`

Logical or slice elements with corresponding elements of *v*.

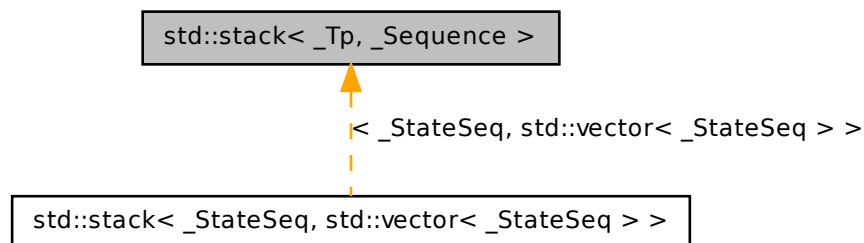
The documentation for this class was generated from the following file:

- [slice_array.h](#)

5.655 std::stack< _Tp, _Sequence > Class Template Reference

A standard container giving FILO behavior.

Inheritance diagram for std::stack< _Tp, _Sequence >:



Public Types

- `typedef _Sequence::const_reference` **const_reference**
- `typedef _Sequence` **container_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size_type**
- `typedef _Sequence::value_type` **value_type**

Public Member Functions

- [stack](#) (const _Sequence &__c)

- **stack** (_Sequence &&__c=_Sequence())
- template<typename... _Args>
void **emplace** (_Args &&...__args)
- bool **empty** () const
- void **pop** ()
- void **push** (value_type &&__x)
- void **push** (const value_type &__x)
- size_type **size** () const
- void **swap** (stack &__s)
- const_reference **top** () const
- reference **top** ()

Protected Attributes

- _Sequence **c**

Friends

- template<typename _Tp1, typename _Seq1 >
bool **operator**< (const stack<_Tp1, _Seq1 > &, const stack<_Tp1, _Seq1 > &)
- template<typename _Tp1, typename _Seq1 >
bool **operator**== (const stack<_Tp1, _Seq1 > &, const stack<_Tp1, _Seq1 > &)

5.655.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>> class std::stack<_Tp, _Sequence >

A standard container giving FILO behavior. Meets many of the requirements of a **container**, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to **std::deque**, but it can be any type that supports **back**, **push_back**, and **pop_front**, such as **std::list**, **std::vector**, or an appropriate user-defined type.

Members not found in *normal* containers are **container_type**, which is a typedef for the second Sequence parameter, and **push**, **pop**, and **top**, which are standard stack/FILO operations.

Definition at line 92 of file stl_stack.h.

5.655.2 Constructor & Destructor Documentation

5.655.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
std::stack< _Tp, _Sequence >::stack (const _Sequence & __c)
[inline, explicit]`

Default constructor creates no elements.

Definition at line 130 of file stl_stack.h.

5.655.3 Member Function Documentation

5.655.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack< _Tp, _Sequence >::empty () const [inline]`

Returns true if the stack is empty.

Definition at line 142 of file stl_stack.h.

5.655.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::pop () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop\(\)](#) is called.

Definition at line 208 of file stl_stack.h.

5.655.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::push (const value_type & __x)
[inline]`

Add data to the top of the stack.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 182 of file `stl_stack.h`.

5.655.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack<_Tp, _Sequence>::size () const [inline]`

Returns the number of elements in the stack.

Definition at line 147 of file `stl_stack.h`.

5.655.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack<_Tp, _Sequence>::top () [inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 155 of file `stl_stack.h`.

5.655.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack<_Tp, _Sequence>::top () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 166 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl_stack.h](#)

5.656 `std::student_t_distribution<_RealType>` Class Template Reference

A [student_t_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **student_t_distribution** (_RealType __n=_RealType(1))
- **student_t_distribution** (const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void [reset](#) ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & **operator<<** ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::student_t_distribution](#)< _RealType1 > &)
- template<typename _RealType1 >
bool **operator==** (const [std::student_t_distribution](#)< _RealType1 > &__d1, const [std::student_t_distribution](#)< _RealType1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator>>** ([std::basic_istream](#)< _CharT, _Traits > &, [std::student_t_distribution](#)< _RealType1 > &)

5.656.1 Detailed Description

template<typename _RealType = double> class std::student_t_distribution< _RealType >

A [student_t_distribution](#) random number distribution. The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3061 of file random.h.

5.656.2 Member Typedef Documentation

5.656.2.1 `template<typename _RealType = double> typedef _RealType std::student_t_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 3068 of file random.h.

5.656.3 Member Function Documentation

5.656.3.1 `template<typename _RealType = double> result_type std::student_t_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3144 of file random.h.

5.656.3.2 `template<typename _RealType = double> result_type std::student_t_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3137 of file random.h.

5.656.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::student_t_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3152 of file random.h.

References `std::sqrt()`.

5.656.3.4 `template<typename _RealType = double> param_type std::student_t_distribution< _RealType >::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3122 of file random.h.

5.656.3.5 `template<typename _RealType = double> void
std::student_t_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3130 of file random.h.

5.656.3.6 `template<typename _RealType = double> void
std::student_t_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 3105 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`, and `std::normal_distribution< _RealType >::reset()`.

5.656.4 Friends And Related Function Documentation

5.656.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::student_t_distribution< _RealType1 > &) [friend]`

Inserts a student_t_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A student_t_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.656.4.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::student_t_distribution<
_RealType1 > & __d1, const std::student_t_distribution<
_RealType1 > & __d2) [friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3174 of file random.h.

5.656.4.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::student_t_distribution< _RealType1 > &) [friend]`

Extracts a student_t_distribution random number distribution __x from the input stream __is.

Parameters

`__is` An input stream.

`__x` A student_t_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.657 `std::student_t_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef `student_t_distribution< _RealType > distribution_type`

Public Member Functions

- `param_type (_RealType __n=_RealType(1))`
- `_RealType n () const`

Friends

- `bool operator==(const param_type &__p1, const param_type &__p2)`

5.657.1 Detailed Description

`template<typename _RealType = double> struct std::student_t_distribution<_RealType>::param_type`

Parameter type.

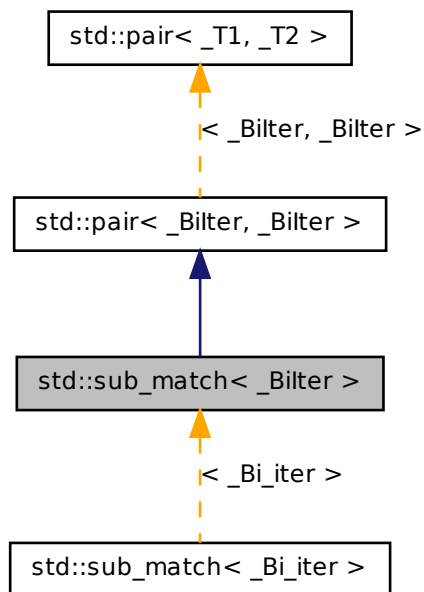
Definition at line 3070 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.658 `std::sub_match<_Biter>` Class Template Reference

Inheritance diagram for `std::sub_match<_Biter>`:



Public Types

- typedef `iterator_traits<_Biter>::difference_type` **difference_type**
- typedef `_Biter` **first_type**
- typedef `_Biter` **iterator**
- typedef `_Biter` **second_type**
- typedef `std::basic_string<value_type>` **string_type**
- typedef `iterator_traits<_Biter>::value_type` **value_type**

Public Member Functions

- int [compare](#) (const [sub_match](#) &__s) const
- int [compare](#) (const [string_type](#) &__s) const
- int [compare](#) (const value_type *__s) const
- difference_type [length](#) () const
- [operator string_type](#) () const
- [string_type](#) [str](#) () const
- void [swap](#) ([pair](#) &__p)

Public Attributes

- _BIter [first](#)
- bool [matched](#)
- _BIter [second](#)

5.658.1 Detailed Description

template<typename _BIter> class std::sub_match<_BIter>

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with [std::basic_string](#) objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 756 of file `regex.h`.

5.658.2 Member Typedef Documentation

5.658.2.1 typedef _BIter std::pair<_BIter, _BIter>::first_type
[inherited]

`first_type` is the first bound type

Definition at line 83 of file `stl_pair.h`.

5.658.2.2 typedef _BIter std::pair<_BIter, _BIter>::second_type
[inherited]

`second_type` is the second bound type

Definition at line 84 of file stl_pair.h.

5.658.3 Member Function Documentation

5.658.3.1 `template<typename _BiIter> int std::sub_match<_BiIter>::compare (const sub_match<_BiIter> & __s) const [inline]`

Compares this and another matched sequence.

Parameters

s Another matched sequence to compare to this one.

Return values

`<0` this matched sequence will collate before *s*.

`=0` this matched sequence is equivalent to *s*.

`<0` this matched sequence will collate after *s*.

Definition at line 815 of file regex.h.

Referenced by `std::operator!=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

5.658.3.2 `template<typename _BiIter> int std::sub_match<_BiIter>::compare (const string_type & __s) const [inline]`

Compares this [sub_match](#) to a string.

Parameters

s A string to compare to this [sub_match](#).

Return values

`<0` this matched sequence will collate before *s*.

`=0` this matched sequence is equivalent to *s*.

`<0` this matched sequence will collate after *s*.

Definition at line 828 of file regex.h.

5.658.3.3 `template<typename _BIter> int std::sub_match<_BIter>::compare (const value_type * __s) const [inline]`

Compares this [sub_match](#) to a C-style string.

Parameters

s A C-style string to compare to this [sub_match](#).

Return values

`<0` this matched sequence will collate before *s*.

`=0` this matched sequence is equivalent to *s*.

`>0` this matched sequence will collate after *s*.

Definition at line 841 of file regex.h.

5.658.3.4 `template<typename _BIter> difference_type std::sub_match<_BIter>::length () const [inline]`

Gets the length of the matching sequence.

Definition at line 772 of file regex.h.

5.658.3.5 `template<typename _BIter> std::sub_match<_BIter>::operator string_type () const [inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the [str\(\)](#) member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 785 of file regex.h.

5.658.3.6 `template<typename _BIter> string_type std::sub_match<_BIter>::str () const [inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

Definition at line 798 of file `regex.h`.

Referenced by `std::sub_match< _Bi_iter >::compare()`, `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

5.658.4 Member Data Documentation

5.658.4.1 `_BiIter std::pair< _BiIter , _BiIter >::first` **[inherited]**

`first` is a copy of the first object

Definition at line 86 of file `stl_pair.h`.

Referenced by `std::sub_match< _Bi_iter >::length()`, `std::sub_match< _Bi_iter >::operator string_type()`, and `std::sub_match< _Bi_iter >::str()`.

5.658.4.2 `_BiIter std::pair< _BiIter , _BiIter >::second` **[inherited]**

`second` is a copy of the second object

Definition at line 87 of file `stl_pair.h`.

Referenced by `std::sub_match< _Bi_iter >::length()`, `std::sub_match< _Bi_iter >::operator string_type()`, and `std::sub_match< _Bi_iter >::str()`.

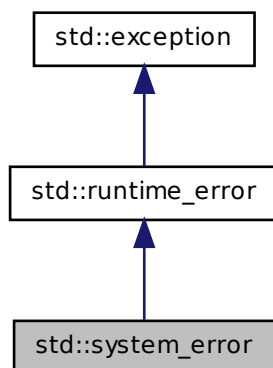
The documentation for this class was generated from the following file:

- [regex.h](#)

5.659 `std::system_error` Class Reference

Thrown to indicate error code of underlying system.

Inheritance diagram for std::system_error:



Public Member Functions

- **system_error** ([error_code](#) __ec=[error_code](#)())
- **system_error** ([error_code](#) __ec, const [string](#) &__what)
- **system_error** (int __v, const [error_category](#) &__ecat, const [string](#) &__what)
- **system_error** (int __v, const [error_category](#) &__ecat)
- const [error_code](#) & **code** () const throw ()
- virtual const char * **what** () const throw ()

5.659.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 307 of file system_error.

5.659.2 Member Function Documentation

5.659.2.1 virtual const char* std::runtime_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [system_error](#)

5.660 std::thread Class Reference

thread

Classes

- class [id](#)
thread::id

Public Types

- typedef [shared_ptr](#)< [_Impl_base](#) > [__shared_base_type](#)
- typedef [__gthread_t](#) [native_handle_type](#)

Public Member Functions

- **thread** (const [thread](#) &)
- template<typename [_Callable](#) >
thread ([_Callable](#) __f)
- template<typename [_Callable](#) , typename... [_Args](#)>
thread ([_Callable](#) &&__f, [_Args](#) &&...__args)
- **thread** ([thread](#) &&__t)
- void **detach** ()
- [thread::id](#) **get_id** () const
- void **join** ()
- bool **joinable** () const
- native_handle_type **native_handle** ()
- [thread](#) & **operator=** (const [thread](#) &)
- [thread](#) & **operator=** ([thread](#) &&__t)
- void **swap** ([thread](#) &__t)

Static Public Member Functions

- static unsigned int **hardware_concurrency** ()

5.660.1 Detailed Description

thread

Definition at line 63 of file thread.

5.660.2 Member Function Documentation

5.660.2.1 native_handle_type std::thread::native_handle () [inline]

Precondition

thread is joinable

Definition at line 179 of file thread.

The documentation for this class was generated from the following file:

- [thread](#)

5.661 std::thread::id Class Reference

[thread::id](#)

Public Member Functions

- **id** (native_handle_type __id)

Friends

- class **hash**< **thread::id** >
- bool **operator**< ([thread::id](#) __x, [thread::id](#) __y)
- template<class _CharT, class _Traits >
[basic_ostream](#)< _CharT, _Traits > & **operator**<< ([basic_ostream](#)< _CharT,
_Traits > &__out, [thread::id](#) __id)
- bool **operator**== ([thread::id](#) __x, [thread::id](#) __y)
- class **thread**

5.661.1 Detailed Description

[thread::id](#)

Definition at line 71 of file thread.

The documentation for this class was generated from the following file:

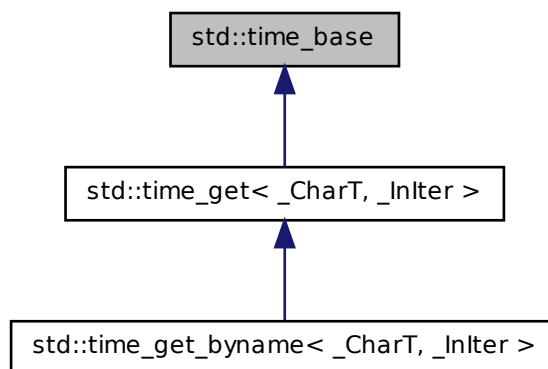
- [thread](#)

5.662 std::time_base Class Reference

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Inheritance diagram for std::time_base:



Public Types

- enum **dateorder** {
 no_order, **dmy**, **mdy**, **ymd**,
 ydm }

5.662.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Definition at line 50 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

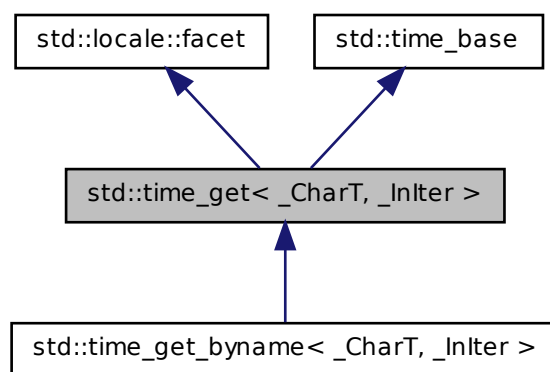
- [locale_facets_nonio.h](#)

5.663 `std::time_get< _CharT, _InIter >` Class Template Reference

Primary class template [time_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

Inheritance diagram for `std::time_get< _CharT, _InIter >`:



Public Types

- typedef [basic_string](#)< `_CharT` > `__string_type`
- enum `dateorder` {
 `no_order`, `dmy`, `mdy`, `ymd`,

ydm }

- typedef `_CharT` [char_type](#)
- typedef `_InIter` [iter_type](#)

Public Member Functions

- [time_get](#) (size_t __refs=0)
- `dateorder` [date_order](#) () const
- [iter_type](#) [get_date](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type](#) [get_monthname](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type](#) [get_time](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type](#) [get_weekday](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type](#) [get_year](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const

Static Public Attributes

- static [locale::id](#) `id`

Protected Member Functions

- virtual `~time_get` ()
- [iter_type](#) [_M_extract_name](#) ([iter_type](#) __beg, [iter_type](#) __end, int &__member, const `_CharT` **__names, size_t __indexlen, [ios_base](#) &__io, [ios_base::iostate](#) &__err) const
- [iter_type](#) [_M_extract_num](#) ([iter_type](#) __beg, [iter_type](#) __end, int &__member, int __min, int __max, size_t __len, [ios_base](#) &__io, [ios_base::iostate](#) &__err) const
- [iter_type](#) [_M_extract_via_format](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm, const `_CharT` *__format) const
- [iter_type](#) [_M_extract_wday_or_month](#) ([iter_type](#) __beg, [iter_type](#) __end, int &__member, const `_CharT` **__names, size_t __indexlen, [ios_base](#) &__io, [ios_base::iostate](#) &__err) const
- virtual `dateorder` [do_date_order](#) () const
- virtual [iter_type](#) [do_get_date](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- virtual [iter_type](#) [do_get_monthname](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__, [ios_base::iostate](#) &__err, tm *__tm) const

- virtual [iter_type](#) [do_get_time](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) & __io, [ios_base::iostate](#) & __err, tm * __tm) const
- virtual [iter_type](#) [do_get_weekday](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) & __io, [ios_base::iostate](#) & __err, tm * __tm) const
- virtual [iter_type](#) [do_get_year](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) & __io, [ios_base::iostate](#) & __err, tm * __tm) const

Static Protected Member Functions

- static [__c_locale](#) [_S_clone_c_locale](#) ([__c_locale](#) & __cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) & __cloc, const char * __s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) & __cloc)
- static [__c_locale](#) [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale](#) [_S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char * __s)

Friends

- class [locale::_Impl](#)

5.663.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get< _CharT, _InIter >`

Primary class template [time_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators. The [time_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time_get](#) facet.

Definition at line 363 of file `locale_facets_nonio.h`.

5.663.2 Member Typedef Documentation

5.663.2.1 `template<typename _CharT, typename _InIter> typedef _CharT std::time_get< _CharT, _InIter >::char_type`

Public typedefs.

Reimplemented in [std::time_get_byname<_CharT, _InIter>](#).

Definition at line 369 of file locale_facets_nonio.h.

5.663.2.2 `template<typename _CharT, typename _InIter> typedef _InIter
std::time_get<_CharT, _InIter>::iter_type`

Public typedefs.

Reimplemented in [std::time_get_byname<_CharT, _InIter>](#).

Definition at line 370 of file locale_facets_nonio.h.

5.663.3 Constructor & Destructor Documentation

5.663.3.1 `template<typename _CharT, typename _InIter> std::time_get<
_CharT, _InIter>::time_get(size_t __refs = 0) [inline,
explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 385 of file locale_facets_nonio.h.

5.663.3.2 `template<typename _CharT, typename _InIter> virtual
std::time_get<_CharT, _InIter>::~~time_get() [inline,
protected, virtual]`

Destructor.

Definition at line 541 of file locale_facets_nonio.h.

5.663.4 Member Function Documentation

5.663.4.1 `template<typename _CharT, typename _InIter> dateorder
std::time_get<_CharT, _InIter>::date_order() const [inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to [time_put::put\(\)](#) only uses month, day, and year. If

the format *x* for the associated locale uses other fields, this function returns timebase::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

Returns

A member of timebase::dateorder.

Definition at line 402 of file locale_facets_nonio.h.

```
5.663.4.2 template<typename _CharT, typename _InIter >
            _GLIBCXX_END_LDBL_NAMESPACE time_base::dateorder
            std::time_get< _CharT, _InIter >::do_date_order ( ) const
            [protected, virtual]
```

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format *x* given to [time_put::put\(\)](#) only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of timebase::dateorder.

Definition at line 618 of file locale_facets_nonio.tcc.

```
5.663.4.3 template<typename _CharT, typename _InIter > _InIter
            std::time_get< _CharT, _InIter >::do_get_date ( iter_type __beg,
            iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
            __tm ) const [protected, virtual]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_date\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 1044 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

5.663.4.4 `template<typename _CharT, typename _InIter> _InIter
 std::time_get<_CharT, _InIter>::do_get_monthname (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
 tm * __tm) const [protected, virtual]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_monthname\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1089 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

5.663.4.5 `template<typename _CharT, typename _InIter> _InIter
std::time_get<_CharT, _InIter>::do_get_time (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual]`

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_time\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 1027 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

5.663.4.6 `template<typename _CharT, typename _InIter> _InIter
std::time_get<_CharT, _InIter>::do_get_weekday (iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm) const [protected, virtual]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_weekday\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 1061 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

5.663.4.7 `template<typename _CharT, typename _InIter> _InIter
 std::time_get<_CharT, _InIter>::do_get_year (iter_type __beg,
 iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
 __tm) const [protected, virtual]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_year\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 1117 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

```
5.663.4.8 template<typename _CharT, typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_date ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [inline]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_date\(\)](#).

If there is a valid date string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 451 of file locale_facets_nonio.h.

```
5.663.4.9 template<typename _CharT, typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_monthname ( iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm ) const [inline]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_monthname\(\)](#).

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 508 of file locale_facets_nonio.h.

```
5.663.4.10 template<typename _CharT , typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_time ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [inline]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_time\(\)](#).

If there is a valid time string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 426 of file locale_facets_nonio.h.

5.663.4.11 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_weekday (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_weekday\(\)](#).

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 479 of file locale_facets_nonio.h.

5.663.4.12 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_year (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_year\(\)](#).

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 534 of file locale_facets_nonio.h.

5.663.5 Member Data Documentation

5.663.5.1 `template<typename _CharT, typename _InIter > locale::id std::time_get< _CharT, _InIter >::id [static]`

Numpunct facet id.

Definition at line 375 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

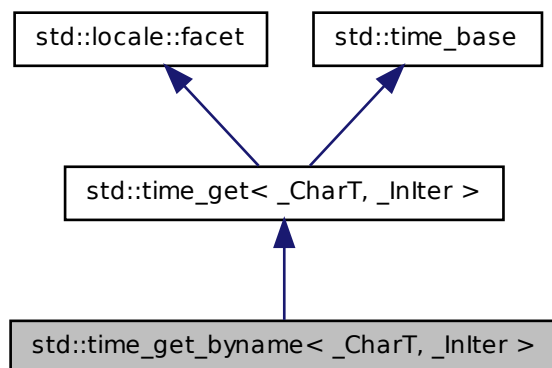
- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.664 `std::time_get_byname< _CharT, _InIter > Class` Template Reference

class [time_get_byname](#) [22.2.5.2].

5.664 std::time_get_byname< _CharT, _InIter > Class Template Reference 3053

Inheritance diagram for std::time_get_byname< _CharT, _InIter >:



Public Types

- typedef `basic_string< _CharT >` `__string_type`
- typedef `_CharT` `char_type`
- enum `dateorder` {
 `no_order`, `dmy`, `mdy`, `ymd`,
 `ydm` }
- typedef `_InIter` `iter_type`

Public Member Functions

- `time_get_byname` (`const char *`, `size_t __refs=0`)
- `dateorder date_order` () `const`
- `iter_type get_date` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_monthname` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_time` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_weekday` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`

- `iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

Static Public Attributes

- static `locale::id id`

Protected Member Functions

- `iter_type _M_extract_name (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type _M_extract_num (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type _M_extract_via_format (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const`
- `iter_type _M_extract_wday_or_month (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- virtual `dateorder do_date_order () const`
- virtual `iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.664.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get_byname< _CharT, _InIter >`

class [time_get_byname](#) [22.2.5.2].

Definition at line 681 of file `locale_facets_nonio.h`.

5.664.2 Member Typedef Documentation

5.664.2.1 `template<typename _CharT , typename _InIter > typedef _CharT
std::time_get_byname< _CharT, _InIter >::char_type`

Public typedefs.

Reimplemented from [std::time_get< _CharT, _InIter >](#).

Definition at line 685 of file `locale_facets_nonio.h`.

5.664.2.2 `template<typename _CharT , typename _InIter > typedef _InIter
std::time_get_byname< _CharT, _InIter >::iter_type`

Public typedefs.

Reimplemented from [std::time_get< _CharT, _InIter >](#).

Definition at line 686 of file `locale_facets_nonio.h`.

5.664.3 Member Function Documentation

5.664.3.1 `template<typename _CharT , typename _InIter > dateorder
std::time_get< _CharT, _InIter >::date_order () const
[inline, inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to [time_put::put\(\)](#) only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns noorder at the moment.

Returns

A member of `timebase::dateorder`.

Definition at line 402 of file `locale_facets_nonio.h`.

5.664.3.2 `template<typename _CharT, typename _InIter >
_GLIBCXX_END_LDBL_NAMESPACE time_base::dateorder
std::time_get< _CharT, _InIter >::do_date_order () const
[protected, virtual, inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of `timebase::dateorder`.

Definition at line 618 of file `locale_facets_nonio.tcc`.

5.664.3.3 `template<typename _CharT, typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_date (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual, inherited]`

Parse input date string.

This function parses a date according to the format `X` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

[get_date\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

5.664 std::time_get_byname<_CharT, _InIter> Class Template Reference 3057

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 1044 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

```
5.664.3.4 template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual, inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_monthname\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1089 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

```
5.664.3.5 template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual, inherited]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

[get_time\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 1027 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

5.664.3.6 `template<typename _CharT, typename _InIter> _InIter
 std::time_get<_CharT, _InIter>::do_get_weekday (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
 tm * __tm) const [protected, virtual, inherited]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

[get_weekday\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

5.664 std::time_get_byname<_CharT, _InIter> Class Template Reference 3059

Returns

Iterator to first char beyond weekday name.

Definition at line 1061 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

```
5.664.3.7 template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual, inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_year\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 1117 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

```
5.664.3.8 template<typename _CharT, typename _InIter> iter_type  
std::time_get<_CharT, _InIter>::get_date ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_date\(\)](#).

If there is a valid date string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, *err* |= [ios_base::failbit](#). If parsing reads all the characters, *err* |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 451 of file `locale_facets_nonio.h`.

5.664.3.9 `template<typename _CharT, typename _InIter> iter_type
 std::time_get<_CharT, _InIter>::get_monthname (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
 tm * __tm) const [inline, inherited]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_monthname\(\)](#).

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, *err* |= [ios_base::failbit](#). If parsing reads all the characters, *err* |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.

5.664 std::time_get_byname<_CharT, _InIter> Class Template Reference 3061

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 508 of file locale_facets_nonio.h.

```
5.664.3.10 template<typename _CharT, typename _InIter> iter_type  
std::time_get<_CharT, _InIter>::get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm )const [inline, inherited]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_time\(\)](#).

If there is a valid time string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 426 of file locale_facets_nonio.h.

```
5.664.3.11 template<typename _CharT, typename _InIter> iter_type  
std::time_get<_CharT, _InIter>::get_weekday ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm )const [inline, inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_weekday\(\)](#).

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 479 of file locale_facets_nonio.h.

```
5.664.3.12 template<typename _CharT, typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_year ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [inline, inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 534 of file `locale_facets_nonio.h`.

5.664.4 Member Data Documentation

5.664.4.1 `template<typename _CharT, typename _InIter> locale::id
std::time_get<_CharT, _InIter>::id [static, inherited]`

Numpunct facet id.

Definition at line 375 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

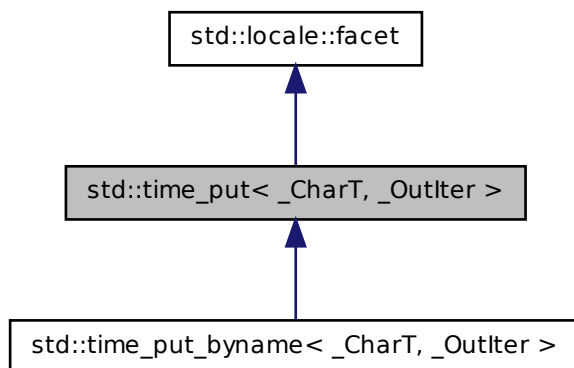
- [locale_facets_nonio.h](#)

5.665 `std::time_put<_CharT, _OutIter>` Class Template Reference

Primary class template [time_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

Inheritance diagram for `std::time_put< _CharT, _OutIter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_OutIter` [iter_type](#)

Public Member Functions

- [time_put](#) (`size_t __refs=0`)
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const `tm * __tm`, `char __format`, `char __mod=0`) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const `tm * __tm`, const `_CharT * __beg`, const `_CharT * __end`) const

Static Public Attributes

- static [locale::id](#) `id`

Protected Member Functions

- virtual [~time_put](#) ()

- virtual [iter_type do_put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.665.1 Detailed Description

template<typename _CharT, typename _OutIter> class std::time_put<_CharT, _OutIter>

Primary class template [time_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by strftime(). The [time_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time_put](#) facet.

Definition at line 710 of file locale_facets_nonio.h.

5.665.2 Member Typedef Documentation

5.665.2.1 **template<typename _CharT, typename _OutIter> typedef _CharT std::time_put<_CharT, _OutIter>::char_type**

Public typedefs.

Reimplemented in [std::time_put_byname<_CharT, _OutIter>](#).

Definition at line 716 of file locale_facets_nonio.h.

5.665.2.2 `template<typename _CharT, typename _OutIter > typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Reimplemented in [std::time_put_byname< _CharT, _OutIter >](#).

Definition at line 717 of file `locale_facets_nonio.h`.

5.665.3 Constructor & Destructor Documentation

5.665.3.1 `template<typename _CharT, typename _OutIter > std::time_put< _CharT, _OutIter >::time_put (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 731 of file `locale_facets_nonio.h`.

5.665.3.2 `template<typename _CharT, typename _OutIter > virtual std::time_put< _CharT, _OutIter >::~~time_put () [inline, protected, virtual]`

Destructor.

Definition at line 777 of file `locale_facets_nonio.h`.

5.665.4 Member Function Documentation

5.665.4.1 `template<typename _CharT, typename _OutIter > _OutIter std::time_put< _CharT, _OutIter >::do_put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod) const [protected, virtual]`

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#) for more details.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
format Format char.
mod Optional modifier char.

Returns

Iterator after writing.

Definition at line 1175 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::time_put< _CharT, _OutIter >::put()`.

5.665.4.2 `template<typename _CharT, typename _OutIter> _OutIter
std::time_put< _CharT, _OutIter >::put (iter_type __s, ios_base &
__io, char_type __fill, const tm * __tm, const _CharT * __beg,
const _CharT * __end) const`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
beg Start of format string.
end End of format string.

Returns

Iterator after writing.

Definition at line 1140 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::time_put<_CharT, _OutIter >::do_put()`, and `std::__ctype_abstract_base<_CharT >::narrow()`.

5.665.4.3 `template<typename _CharT, typename _OutIter > iter_type
std::time_put<_CharT, _OutIter >::put (iter_type __s, ios_base
& __io, char_type __fill, const tm * __tm, char __format, char
__mod = 0) const [inline]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* char_type to use for padding.
- tm* Struct tm with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

Returns

Iterator after writing.

Definition at line 770 of file locale_facets_nonio.h.

References `std::time_put<_CharT, _OutIter >::do_put()`.

5.665.5 Member Data Documentation

5.665.5.1 `template<typename _CharT, typename _OutIter > locale::id
std::time_put<_CharT, _OutIter >::id [static]`

Numpunct facet id.

Definition at line 721 of file locale_facets_nonio.h.

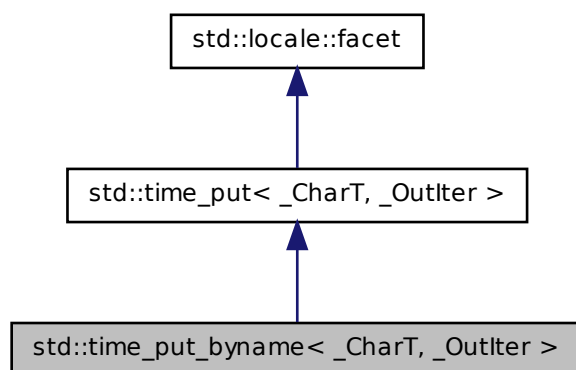
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.666 std::time_put_byname< _CharT, _OutIter > Class Template Reference

class [time_put_byname](#) [22.2.5.4].

Inheritance diagram for std::time_put_byname< _CharT, _OutIter >:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_OutIter` [iter_type](#)

Public Member Functions

- **time_put_byname** (const char *, size_t __refs=0)
- [iter_type](#) **put** ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, char __format, char __mod=0) const
- [iter_type](#) **put** ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, const `_CharT` *__beg, const `_CharT` *__end) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [iter_type](#) [do_put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const [tm](#) *__tm, [char](#) __format, [char](#) __mod) const

Static Protected Member Functions

- static [__c_locale](#) [_S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const [char](#) *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale](#) [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const [char](#) * [_S_get_c_name](#) () throw ()
- static [__c_locale](#) [_S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const [char](#) *__s)

Friends

- class [locale::Impl](#)

5.666.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::time_put_byname<_CharT, _OutIter >`

class [time_put_byname](#) [22.2.5.4].

Definition at line 806 of file [locale_facets_nonio.h](#).

5.666.2 Member Typedef Documentation

5.666.2.1 `template<typename _CharT , typename _OutIter > typedef _CharT std::time_put_byname< _CharT, _OutIter >::char_type`

Public typedefs.

Reimplemented from [std::time_put< _CharT, _OutIter >](#).

Definition at line 810 of file [locale_facets_nonio.h](#).

5.666.2.2 `template<typename _CharT , typename _OutIter > typedef _OutIter std::time_put_byname< _CharT, _OutIter >::iter_type`

Public typedefs.

5.666 std::time_put_byname< _CharT, _OutIter > Class Template Reference 3071

Reimplemented from [std::time_put< _CharT, _OutIter >](#).

Definition at line 811 of file locale_facets_nonio.h.

5.666.3 Member Function Documentation

5.666.3.1 `template<typename _CharT, typename _OutIter > _OutIter
std::time_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, const tm * __tm, char __format,
char __mod) const [protected, virtual, inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#) for more details.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
format Format char.
mod Optional modifier char.

Returns

Iterator after writing.

Definition at line 1175 of file locale_facets_nonio.tcc.

References [std::ios_base::_M_getloc\(\)](#), and [std::__ctype_abstract_base< _CharT >::widen\(\)](#).

Referenced by [std::time_put< _CharT, _OutIter >::put\(\)](#).

5.666.3.2 `template<typename _CharT, typename _OutIter > _OutIter
std::time_put< _CharT, _OutIter >::put (iter_type __s, ios_base &
__io, char_type __fill, const tm * __tm, const _CharT * __beg,
const _CharT * __end) const [inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
beg Start of format string.
end End of format string.

Returns

Iterator after writing.

Definition at line 1140 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::time_put< _CharT, _OutIter >::do_put(), and std::__ctype_abstract_base< _CharT >::narrow().

5.666.3.3 `template<typename _CharT, typename _OutIter> iter_type
 std::time_put< _CharT, _OutIter >::put (iter_type __s, ios_base
 & __io, char_type __fill, const tm * __tm, char __format, char
 __mod = 0) const [inline, inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning [time_put::do_put\(\)](#).

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
format Format char.
mod Optional modifier char.

Returns

Iterator after writing.

Definition at line 770 of file locale_facets_nonio.h.

References std::time_put< _CharT, _OutIter >::do_put().

5.666.4 Member Data Documentation

5.666.4.1 `template<typename _CharT, typename _OutIter> locale::id
std::time_put<_CharT, _OutIter>::id [static, inherited]`

Numpunct facet id.

Definition at line 721 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

5.667 std::timed_mutex Class Reference

[timed_mutex](#)

Public Types

- typedef `__native_type * native_handle_type`

Public Member Functions

- **timed_mutex** (const [timed_mutex](#) &)
- void **lock** ()
- native_handle_type **native_handle** ()
- [timed_mutex](#) & **operator=** (const [timed_mutex](#) &)
- bool **try_lock** ()
- template<class _Rep, class _Period>
bool **try_lock_for** (const [chrono::duration](#)<_Rep, _Period> &__rtime)
- template<class _Clock, class _Duration>
bool **try_lock_until** (const [chrono::time_point](#)<_Clock, _Duration> &__-
atime)
- void **unlock** ()

5.667.1 Detailed Description

[timed_mutex](#)

Definition at line 166 of file mutex.

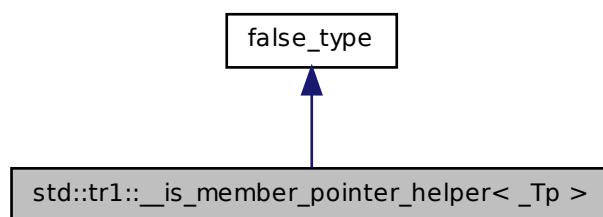
The documentation for this class was generated from the following file:

- [mutex](#)

5.668 `std::tr1::__is_member_pointer_helper< _Tp >` Struct Template Reference

`is_member_pointer`

Inheritance diagram for `std::tr1::__is_member_pointer_helper< _Tp >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > `type`
- typedef `_Tp` `value_type`

Static Public Attributes

- static const `_Tp` `value`

5.668.1 Detailed Description

`template<typename _Tp> struct std::tr1::__is_member_pointer_helper< _Tp >`

`is_member_pointer`

Definition at line 295 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.669 `std::tr1::add_const< _Tp >` Struct Template Reference

[add_const](#)

Public Types

- typedef `_Tp const` **type**

5.669.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_const< _Tp >
```

[add_const](#)

Definition at line 426 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.670 `std::tr1::add_cv< _Tp >` Struct Template Reference

[add_cv](#)

Public Types

- typedef `add_const< typename add_volatile< _Tp >::type >::type` **type**

5.670.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_cv< _Tp >
```

[add_cv](#)

Definition at line 436 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.671 `std::tr1::add_pointer< _Tp >` Struct Template Reference

[add_pointer](#)

Public Types

- typedef [remove_reference< _Tp >::type](#) * **type**

5.671.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_pointer< _Tp >
```

[add_pointer](#)

Definition at line 491 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.672 `std::tr1::add_volatile< _Tp >` Struct Template Reference

[add_volatile](#)

Public Types

- typedef `_Tp volatile` **type**

5.672.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_volatile< _Tp >
```

[add_volatile](#)

Definition at line 431 of file `tr1_impl/type_traits`.

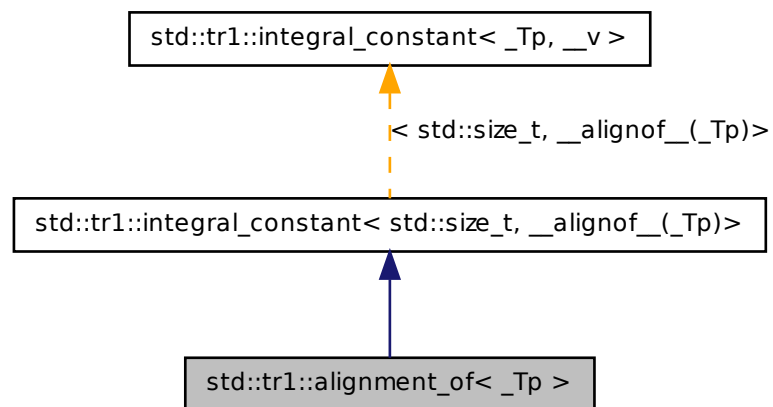
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.673 std::tr1::alignment_of< _Tp > Struct Template Reference

[alignment_of](#)

Inheritance diagram for std::tr1::alignment_of< _Tp >:



Public Types

- typedef [integral_constant](#)< std::size_t, __v > **type**
- typedef std::size_t **value_type**

Static Public Attributes

- static const std::size_t **value**

5.673.1 Detailed Description

```
template<typename _Tp> struct std::tr1::alignment_of< _Tp >
```

[alignment_of](#)

Definition at line 350 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.674 `std::tr1::array< _Tp, _Nm >` Struct Template Reference

A standard container for storing a fixed size sequence of elements.

Public Types

- typedef const value_type * **const_iterator**
- typedef const _Tp * **const_pointer**
- typedef const value_type & **const_reference**
- typedef [std::reverse_iterator](#)< const_iterator > **const_reverse_iterator**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **iterator**
- typedef _Tp * **pointer**
- typedef value_type & **reference**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- reference **at** (size_type __n)
- const_reference **at** (size_type __n) const
- reference **back** ()
- const_reference **back** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- const _Tp * **data** () const
- _Tp * **data** ()
- bool **empty** () const
- iterator **end** ()

- const_iterator **end** () const
- void **fill** (const value_type &__u)
- reference **front** ()
- const_reference **front** () const
- size_type **max_size** () const
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- [const_reverse_iterator](#) **rbegin** () const
- [reverse_iterator](#) **rbegin** ()
- [reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- size_type **size** () const
- void **swap** ([array](#) &__other)

Public Attributes

- value_type **_M_instance** [_Nm?_Nm:1]

5.674.1 Detailed Description

template<typename _Tp, std::size_t _Nm> struct std::tr1::array< _Tp, _Nm >

A standard container for storing a fixed size sequence of elements. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

Parameters

Tp Type of element. Required to be a complete type.

N Number of elements.

Definition at line 49 of file tr1_impl/array.

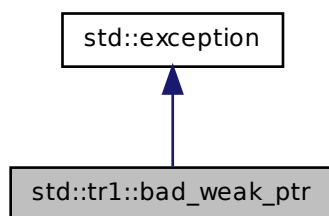
The documentation for this struct was generated from the following file:

- [tr1_impl/array](#)

5.675 std::tr1::bad_weak_ptr Class Reference

Exception possibly thrown by [shared_ptr](#).

Inheritance diagram for `std::tr1::bad_weak_ptr`:



Public Member Functions

- virtual char const * [what](#) () const throw ()

5.675.1 Detailed Description

Exception possibly thrown by [shared_ptr](#).

Definition at line 58 of file `boost_sp_counted_base.h`.

5.675.2 Member Function Documentation

5.675.2.1 virtual char const* `std::tr1::bad_weak_ptr::what ()` const throw () [inline, virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

Definition at line 62 of file `boost_sp_counted_base.h`.

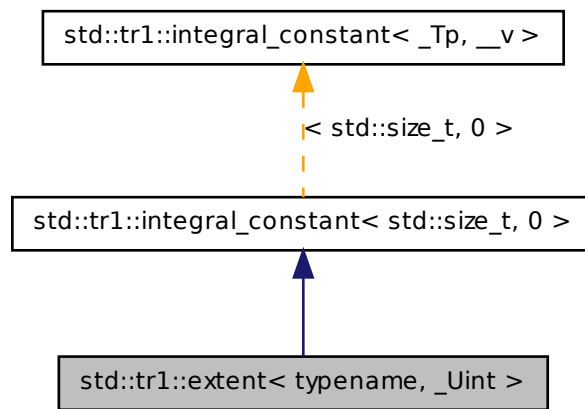
The documentation for this class was generated from the following file:

- [boost_sp_counted_base.h](#)

5.676 `std::tr1::extent< typename, _Uint >` Struct Template Reference

extent

Inheritance diagram for `std::tr1::extent< typename, _Uint >`:



Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value_type**

Static Public Attributes

- static const `std::size_t` **value**

5.676.1 Detailed Description

`template<typename, unsigned _Uint = 0> struct std::tr1::extent< typename, _-
_Uint >`

extent

Definition at line 368 of file tr1_impl/type_traits.

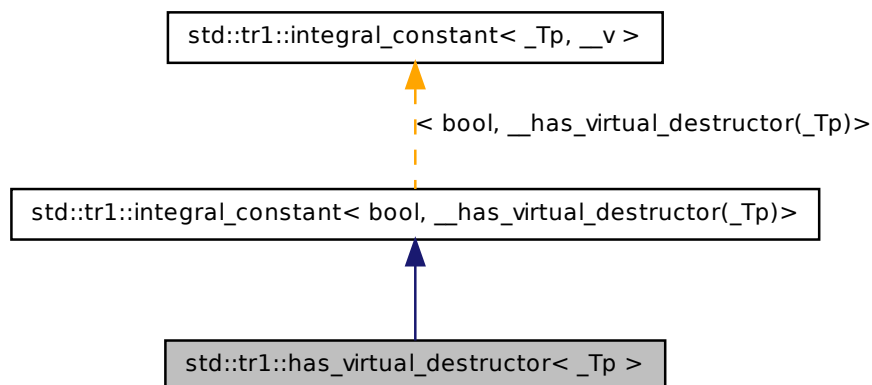
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.677 `std::tr1::has_virtual_destructor< _Tp >` Struct Template Reference

[has_virtual_destructor](#)

Inheritance diagram for `std::tr1::has_virtual_destructor< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.677.1 Detailed Description

```
template<typename _Tp> struct std::tr1::has_virtual_destructor< _Tp >
```

[has_virtual_destructor](#)

Definition at line 344 of file tr1_impl/type_traits.

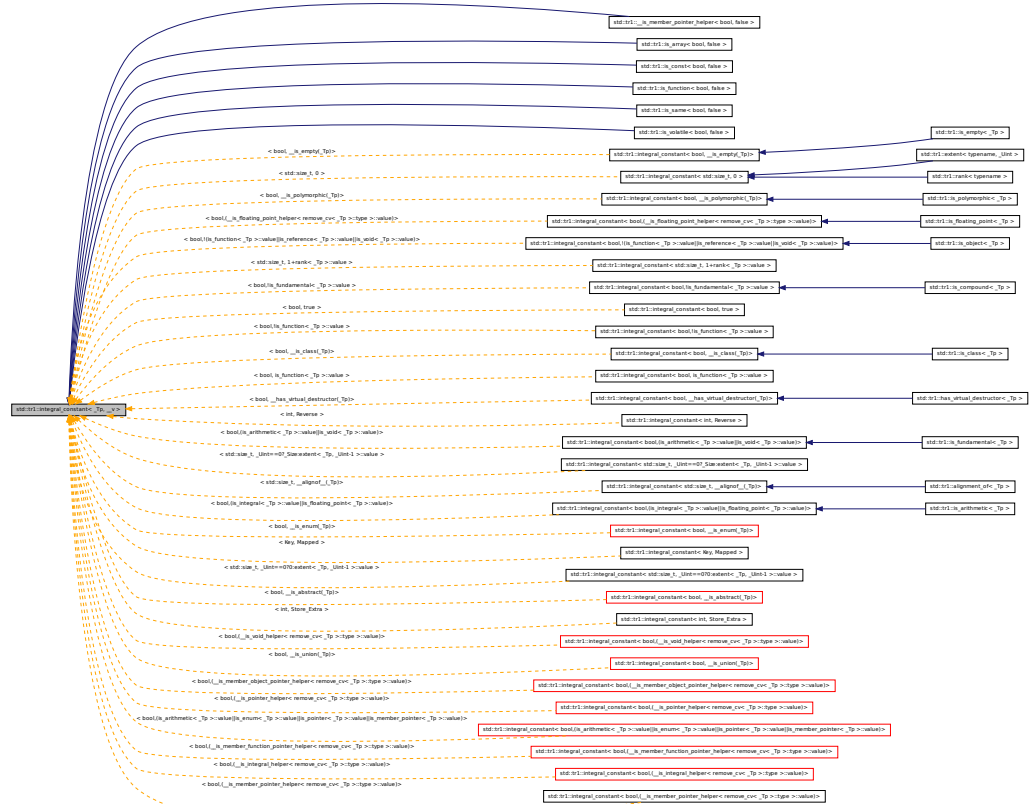
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.678 std::tr1::integral_constant< _Tp, __v > Struct Template Reference

[integral_constant](#)

Inheritance diagram for `std::tr1::integral_constant< _Tp, __v >`:



Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value_type**

Static Public Attributes

- static const `_Tp` **value**

5.678.1 Detailed Description

```
template<typename _Tp, _Tp __v> struct std::tr1::integral_constant< _Tp, __v
>
```

[integral_constant](#)

Definition at line 67 of file tr1_impl/type_traits.

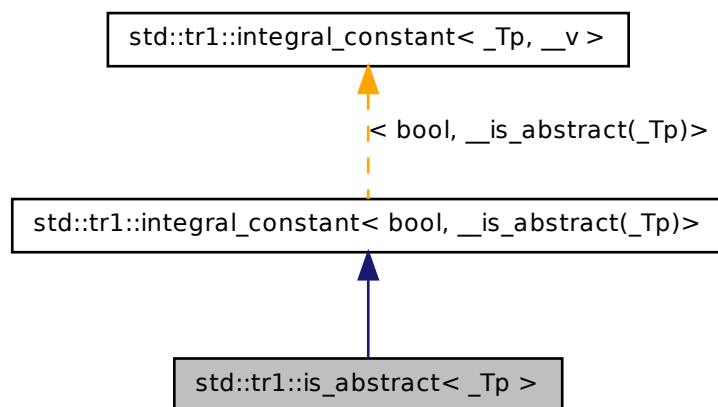
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.679 std::tr1::is_abstract< _Tp > Struct Template Reference

[is_abstract](#)

Inheritance diagram for std::tr1::is_abstract< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**

- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.679.1 Detailed Description

template<typename _Tp> struct std::tr1::is_abstract< _Tp >

[is_abstract](#)

Definition at line 338 of file tr1_impl/type_traits.

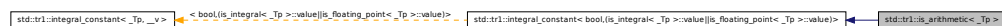
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.680 std::tr1::is_arithmetic< _Tp > Struct Template Reference

[is_arithmetic](#)

Inheritance diagram for std::tr1::is_arithmetic< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.680.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_arithmetic< _Tp >
```

[is_arithmetic](#)

Definition at line 255 of file tr1_impl/type_traits.

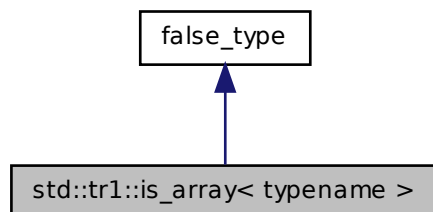
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.681 std::tr1::is_array< typename > Struct Template Reference

[is_array](#)

Inheritance diagram for std::tr1::is_array< typename >:



Public Types

- typedef [integral_constant](#)< _Tp, __v > **type**
- typedef _Tp **value_type**

Static Public Attributes

- static const _Tp **value**

5.681.1 Detailed Description

`template<typename> struct std::tr1::is_array< typename >`

[is_array](#)

Definition at line 147 of file `tr1_impl/type_traits`.

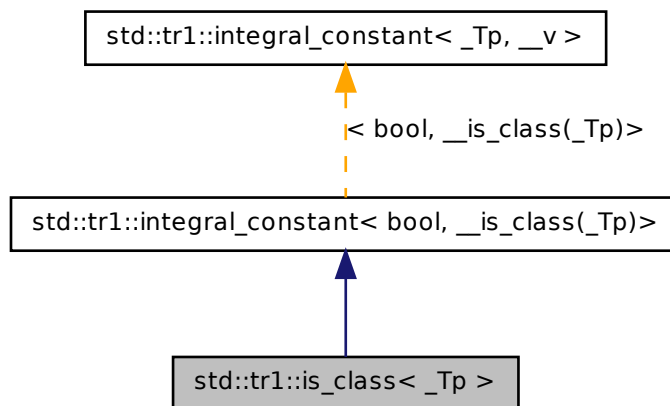
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.682 `std::tr1::is_class< _Tp >` Struct Template Reference

[is_class](#)

Inheritance diagram for `std::tr1::is_class< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.682.1 Detailed Description

template<typename _Tp> struct std::tr1::is_class< _Tp >

[is_class](#)

Definition at line 218 of file tr1_impl/type_traits.

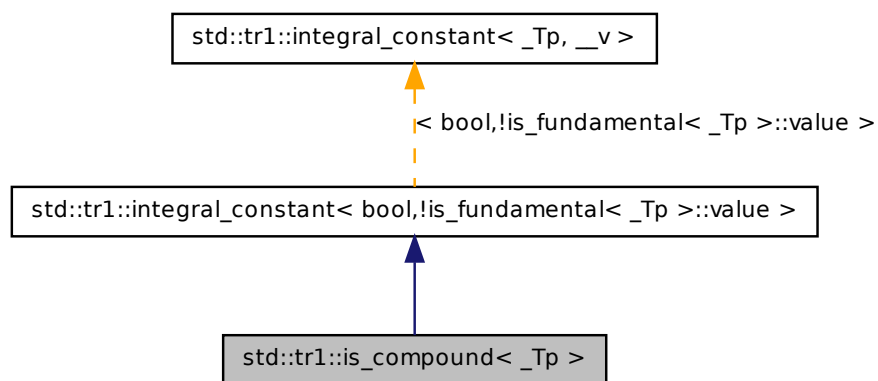
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.683 std::tr1::is_compound< _Tp > Struct Template Reference

[is_compound](#)

Inheritance diagram for std::tr1::is_compound< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.683.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_compound< _Tp >`

[is_compound](#)

Definition at line 290 of file `tr1_impl/type_traits`.

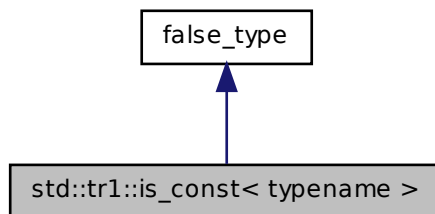
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.684 `std::tr1::is_const< typename >` Struct Template Reference

[is_const](#)

Inheritance diagram for `std::tr1::is_const< typename >`:



Public Types

- typedef [integral_constant](#)< _Tp, __v > **type**
- typedef _Tp **value_type**

Static Public Attributes

- static const _Tp **value**

5.684.1 Detailed Description

template<typename> struct std::tr1::is_const< typename >

[is_const](#)

Definition at line 308 of file tr1_impl/type_traits.

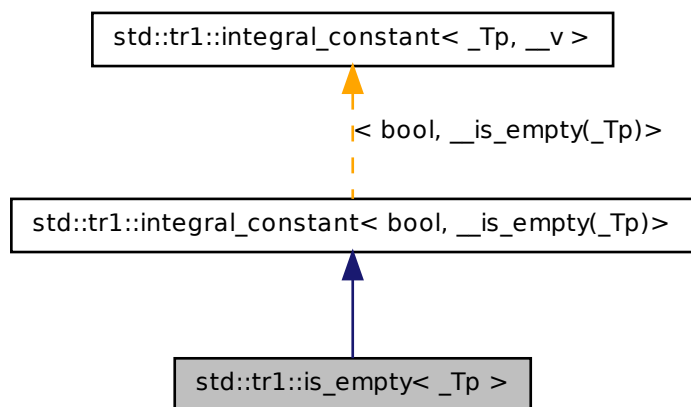
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.685 std::tr1::is_empty< _Tp > Struct Template Reference

[is_empty](#)

Inheritance diagram for `std::tr1::is_empty< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.685.1 Detailed Description

template<typename _Tp> struct `std::tr1::is_empty< _Tp >`

[is_empty](#)

Definition at line 326 of file `tr1_impl/type_traits`.

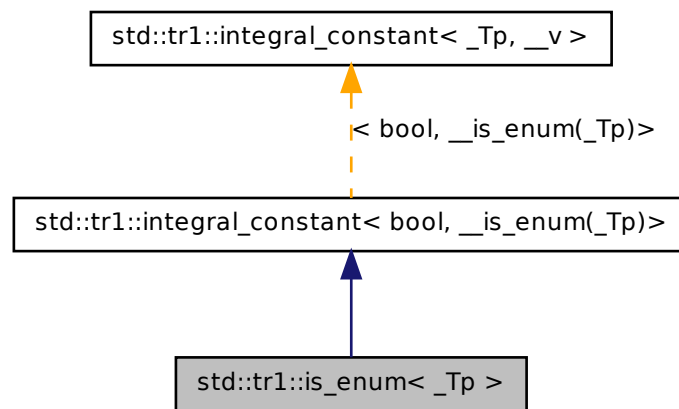
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.686 std::tr1::is_enum< _Tp > Struct Template Reference

[is_enum](#)

Inheritance diagram for std::tr1::is_enum< _Tp >:



Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value_type**

Static Public Attributes

- static const `bool` **value**

5.686.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_enum< _Tp >
```

[is_enum](#)

Definition at line 206 of file tr1_impl/type_traits.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.687 std::tr1::is_floating_point< _Tp > Struct Template Reference

[is_floating_point](#)

Inheritance diagram for std::tr1::is_floating_point< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.687.1 Detailed Description

template<typename _Tp> struct std::tr1::is_floating_point< _Tp >

[is_floating_point](#)

Definition at line 140 of file tr1_impl/type_traits.

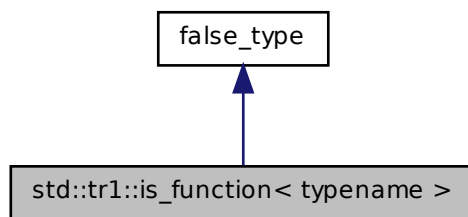
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.688 `std::tr1::is_function< typename >` Struct Template Reference

[is_function](#)

Inheritance diagram for `std::tr1::is_function< typename >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value_type**

Static Public Attributes

- static const `_Tp` **value**

5.688.1 Detailed Description

template<typename> struct `std::tr1::is_function< typename >`

[is_function](#)

Definition at line 224 of file `tr1_impl/type_traits`.

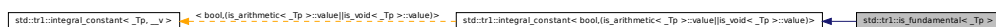
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.689 `std::tr1::is_fundamental< _Tp >` Struct Template Reference

[is_fundamental](#)

Inheritance diagram for `std::tr1::is_fundamental< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.689.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_fundamental< _Tp >`

[is_fundamental](#)

Definition at line 262 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

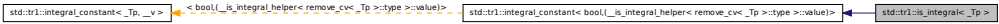
- [tr1_impl/type_traits](#)

5.690 `std::tr1::is_integral< _Tp >` Struct Template Reference

[is_integral](#)

5.691 **std::tr1::is_member_function_pointer< _Tp > Struct Template Reference**

Inheritance diagram for std::tr1::is_integral< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.690.1 Detailed Description

template<typename _Tp> struct std::tr1::is_integral< _Tp >

[is_integral](#)

Definition at line 126 of file tr1_impl/type_traits.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.691 **std::tr1::is_member_function_pointer< _Tp > Struct Template Reference**

[is_member_function_pointer](#)

Inheritance diagram for std::tr1::is_member_function_pointer< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**

- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.691.1 Detailed Description

template<typename _Tp> struct std::tr1::is_member_function_pointer< _Tp >

[is_member_function_pointer](#)

Definition at line 199 of file tr1_impl/type_traits.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.692 std::tr1::is_member_object_pointer< _Tp > Struct Template Reference

[is_member_object_pointer](#)

Inheritance diagram for std::tr1::is_member_object_pointer< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.692.1 Detailed Description

template<typename _Tp> struct std::tr1::is_member_object_pointer< _Tp >

[is_member_object_pointer](#)

Definition at line 186 of file tr1_impl/type_traits.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.693 std::tr1::is_object< _Tp > Struct Template Reference

[is_object](#)

Inheritance diagram for std::tr1::is_object< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.693.1 Detailed Description

template<typename _Tp> struct std::tr1::is_object< _Tp >

[is_object](#)

Definition at line 269 of file tr1_impl/type_traits.

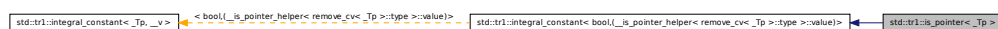
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.694 `std::tr1::is_pointer< _Tp >` Struct Template Reference

[is_pointer](#)

Inheritance diagram for `std::tr1::is_pointer< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.694.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_pointer< _Tp >`

[is_pointer](#)

Definition at line 165 of file `tr1_impl/type_traits`.

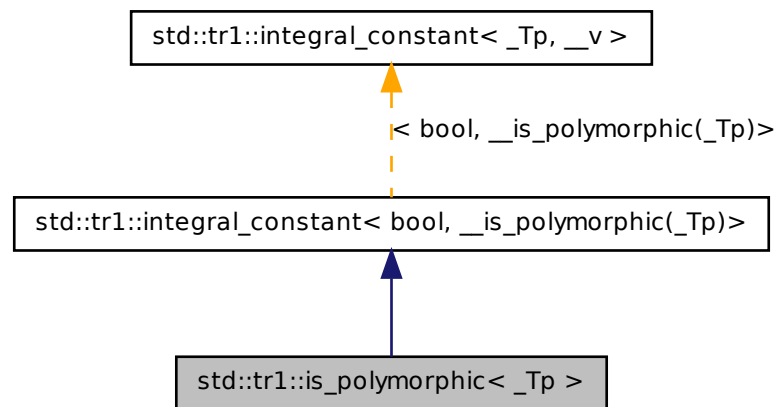
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.695 `std::tr1::is_polymorphic< _Tp >` Struct Template Reference

[is_polymorphic](#)

Inheritance diagram for std::tr1::is_polymorphic< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.695.1 Detailed Description

template<typename _Tp> struct std::tr1::is_polymorphic< _Tp >

[is_polymorphic](#)

Definition at line 332 of file `tr1_impl/type_traits`.

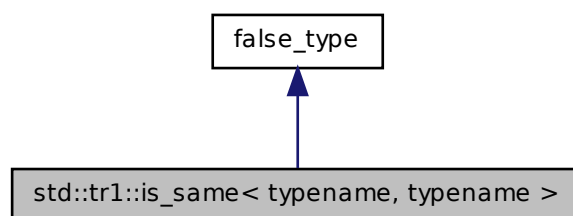
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.696 `std::tr1::is_same< typename, typename >` Struct Template Reference

[is_same](#)

Inheritance diagram for `std::tr1::is_same< typename, typename >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value_type**

Static Public Attributes

- static const `_Tp` **value**

5.696.1 Detailed Description

```
template<typename, typename> struct std::tr1::is_same< typename, typename
>
```

[is_same](#)

Definition at line 389 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.697 std::tr1::is_scalar< _Tp > Struct Template Reference

[is_scalar](#)

Inheritance diagram for std::tr1::is_scalar< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.697.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_scalar< _Tp >
```

[is_scalar](#)

Definition at line 281 of file tr1_impl/type_traits.

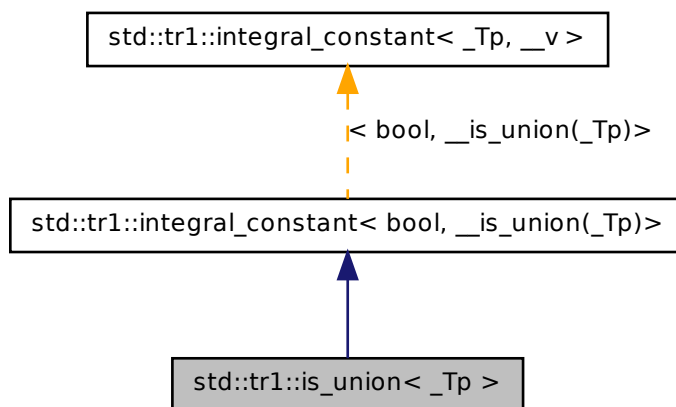
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.698 std::tr1::is_union< _Tp > Struct Template Reference

[is_union](#)

Inheritance diagram for `std::tr1::is_union< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.698.1 Detailed Description

template<typename _Tp> struct `std::tr1::is_union< _Tp >`

[is_union](#)

Definition at line 212 of file `tr1_impl/type_traits`.

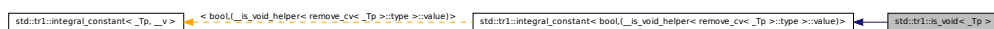
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.699 std::tr1::is_void< _Tp > Struct Template Reference

[is_void](#)

Inheritance diagram for std::tr1::is_void< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.699.1 Detailed Description

template<typename _Tp> struct std::tr1::is_void< _Tp >

[is_void](#)

Definition at line 96 of file tr1_impl/type_traits.

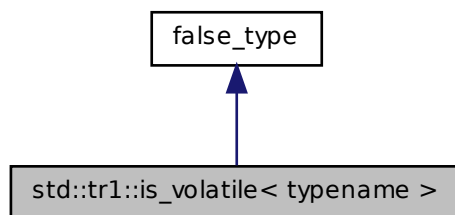
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.700 std::tr1::is_volatile< typename > Struct Template Reference

[is_volatile](#)

Inheritance diagram for `std::tr1::is_volatile< typename >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value_type**

Static Public Attributes

- static const `_Tp` **value**

5.700.1 Detailed Description

`template<typename> struct std::tr1::is_volatile< typename >`

[is_volatile](#)

Definition at line 317 of file `tr1_impl/type_traits`.

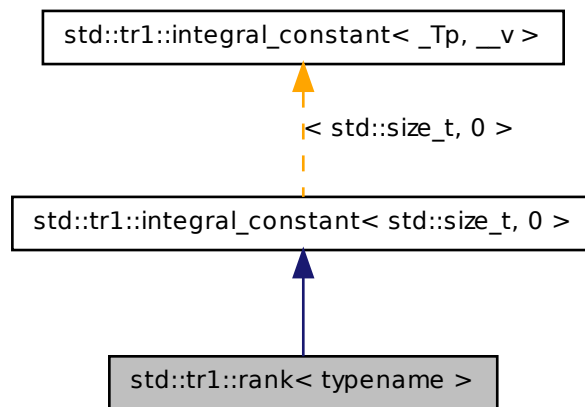
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.701 `std::tr1::rank< typename >` Struct Template Reference

`rank`

Inheritance diagram for std::tr1::rank< typename >:



Public Types

- typedef [integral_constant](#)< std::size_t, __v > **type**
- typedef std::size_t **value_type**

Static Public Attributes

- static const std::size_t **value**

5.701.1 Detailed Description

template<typename> struct std::tr1::rank< typename >

rank

Definition at line 355 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.702 `std::tr1::remove_all_extents< _Tp >` Struct Template Reference

[remove_all_extents](#)

Public Types

- `typedef _Tp type`

5.702.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_all_extents< _Tp >`

[remove_all_extents](#)

Definition at line 459 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.703 `std::tr1::remove_const< _Tp >` Struct Template Reference

[remove_const](#)

Public Types

- `typedef _Tp type`

5.703.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_const< _Tp >`

[remove_const](#)

Definition at line 400 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.704 `std::tr1::remove_cv< _Tp >` Struct Template Reference

[remove_cv](#)

Public Types

- typedef [remove_const](#)< typename [remove_volatile](#)< _Tp >::type >::type **type**

5.704.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_cv< _Tp >`

[remove_cv](#)

Definition at line 418 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.705 `std::tr1::remove_extent< _Tp >` Struct Template Reference

[remove_extent](#)

Public Types

- typedef `_Tp` **type**

5.705.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_extent< _Tp >`

[remove_extent](#)

Definition at line 446 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.706 `std::tr1::remove_pointer< _Tp >` Struct Template Reference

[remove_pointer](#)

Inherits `std::tr1::__remove_pointer_helper< _Tp, remove_cv< _Tp >::type >`.

Public Types

- `typedef _Tp type`

5.706.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_pointer< _Tp >`

[remove_pointer](#)

Definition at line 482 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.707 `std::tr1::remove_volatile< _Tp >` Struct Template Reference

[remove_volatile](#)

Public Types

- `typedef _Tp type`

5.707.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_volatile< _Tp >`

[remove_volatile](#)

Definition at line 409 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.708 std::try_to_lock_t Struct Reference

Try to acquire ownership of the mutex without blocking.

5.708.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

Definition at line 378 of file mutex.

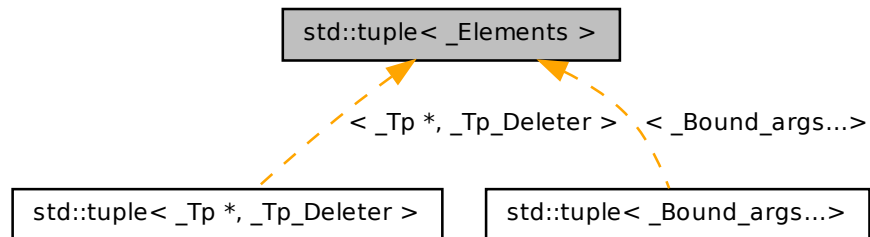
The documentation for this struct was generated from the following file:

- [mutex](#)

5.709 std::tuple< _Elements > Class Template Reference

tuple

Inheritance diagram for std::tuple< _Elements >:



Public Member Functions

- **tuple** (const _Elements &...__elements)
- **tuple** (const [tuple](#) &)
- template<typename... _UElements>
tuple ([tuple](#)< _UElements...> &__in)

- **tuple** (tuple &&__in)
- template<typename... _UElements>
tuple (_UElements &&...__elements)
- template<typename... _UElements>
tuple (const tuple< _UElements...> &__in)
- template<typename... _UElements>
tuple (tuple< _UElements...> &&__in)
- tuple & **operator=** (tuple &&__in)
- template<typename... _UElements>
tuple & **operator=** (const tuple< _UElements...> &__in)
- template<typename... _UElements>
tuple & **operator=** (tuple< _UElements...> &&__in)
- tuple & **operator=** (const tuple &__in)
- void **swap** (tuple &__in)

5.709.1 Detailed Description

template<typename... _Elements> class std::tuple< _Elements >

tuple

Definition at line 224 of file tuple.

The documentation for this class was generated from the following file:

- tuple

5.710 std::tuple< _T1, _T2 > Class Template Reference

tuple (2-element), with construction and assignment from a pair.

Inherits _Tuple_impl< 0, _T1, _T2 >.

Public Member Functions

- **tuple** (const _T1 &__a1, const _T2 &__a2)
- **tuple** (const tuple &)
- template<typename _U1, typename _U2 >
tuple (const pair< _U1, _U2 > &__in)
- template<typename _U1, typename _U2 >
tuple (pair< _U1, _U2 > &&__in)
- **tuple** (tuple &&__in)

- `template<typename _U1, typename _U2 >`
`tuple (_U1 &&__a1, _U2 &&__a2)`
- `template<typename _U1, typename _U2 >`
`tuple (const tuple< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`
`tuple (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >`
`tuple & operator= (const tuple< _U1, _U2 > &__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in)`
- `template<typename _U1, typename _U2 >`
`tuple & operator= (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >`
`tuple & operator= (const pair< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`
`tuple & operator= (pair< _U1, _U2 > &&__in)`
- `void swap (tuple &__in)`

5.710.1 Detailed Description

`template<typename _T1, typename _T2> class std::tuple< _T1, _T2 >`

tuple (2-element), with construction and assignment from a pair.

Definition at line 306 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

5.711 std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference

Public Types

- `typedef _Head type`

5.711.1 Detailed Description

`template<typename _Head, typename... _Tail> struct std::tuple_element< 0, tuple< _Head, _Tail...> >`

Basis case for tuple_element: The first element is the one we're seeking.

Definition at line 419 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.712 `std::tuple_element< __i, tuple< _Head, _Tail...>>` Struct Template Reference

Inherits `tuple_element< __i-1, tuple< _Tail...>>`.

5.712.1 Detailed Description

```
template<std::size_t __i, typename _Head, typename... _Tail> struct std::tuple_
element< __i, tuple< _Head, _Tail...>> >
```

Recursive case for `tuple_element`: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

Definition at line 412 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.713 `std::tuple_size< tuple< _Elements...>>` Struct Template Reference

```
class tuple_size
```

Static Public Attributes

- static const `std::size_t` **value**

5.713.1 Detailed Description

```
template<typename... _Elements> struct std::tuple_size< tuple< _Elements...>>
>
```

```
class tuple_size
```

Definition at line 430 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.714 std::type_info Class Reference

Part of RTTI.

Inherited by `__cxxabiv1::__array_type_info`, `__cxxabiv1::__class_type_info`, `__cxxabiv1::__enum_type_info`, `__cxxabiv1::__function_type_info`, `__cxxabiv1::__fundamental_type_info`, and `__cxxabiv1::__pbase_type_info`.

Public Member Functions

- virtual `~type_info()`
- virtual bool `__do_catch` (const `type_info` * __thr_type, void ** __thr_obj, unsigned __outer) const
- virtual bool `__do_upcast` (const `__cxxabiv1::__class_type_info` * __target, void ** __obj_ptr) const
- virtual bool `__is_function_p` () const
- virtual bool `__is_pointer_p` () const
- bool `before` (const `type_info` & __arg) const
- const char * `name` () const
- bool `operator!=` (const `type_info` & __arg) const
- bool `operator==` (const `type_info` & __arg) const

Protected Member Functions

- `type_info` (const char * __n)

Protected Attributes

- const char * `__name`

5.714.1 Detailed Description

Part of RTTI. The `type_info` class describes type information generated by an implementation.

Definition at line 87 of file typeinfo.

5.714.2 Constructor & Destructor Documentation

5.714.2.1 virtual std::type_info::~~type_info () [virtual]

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated [type_info](#) structures in the new-abi.

5.714.3 Member Function Documentation

5.714.3.1 const char* std::type_info::name () const [inline]

Returns an *implementation-defined* byte string; this is not portable between compilers!

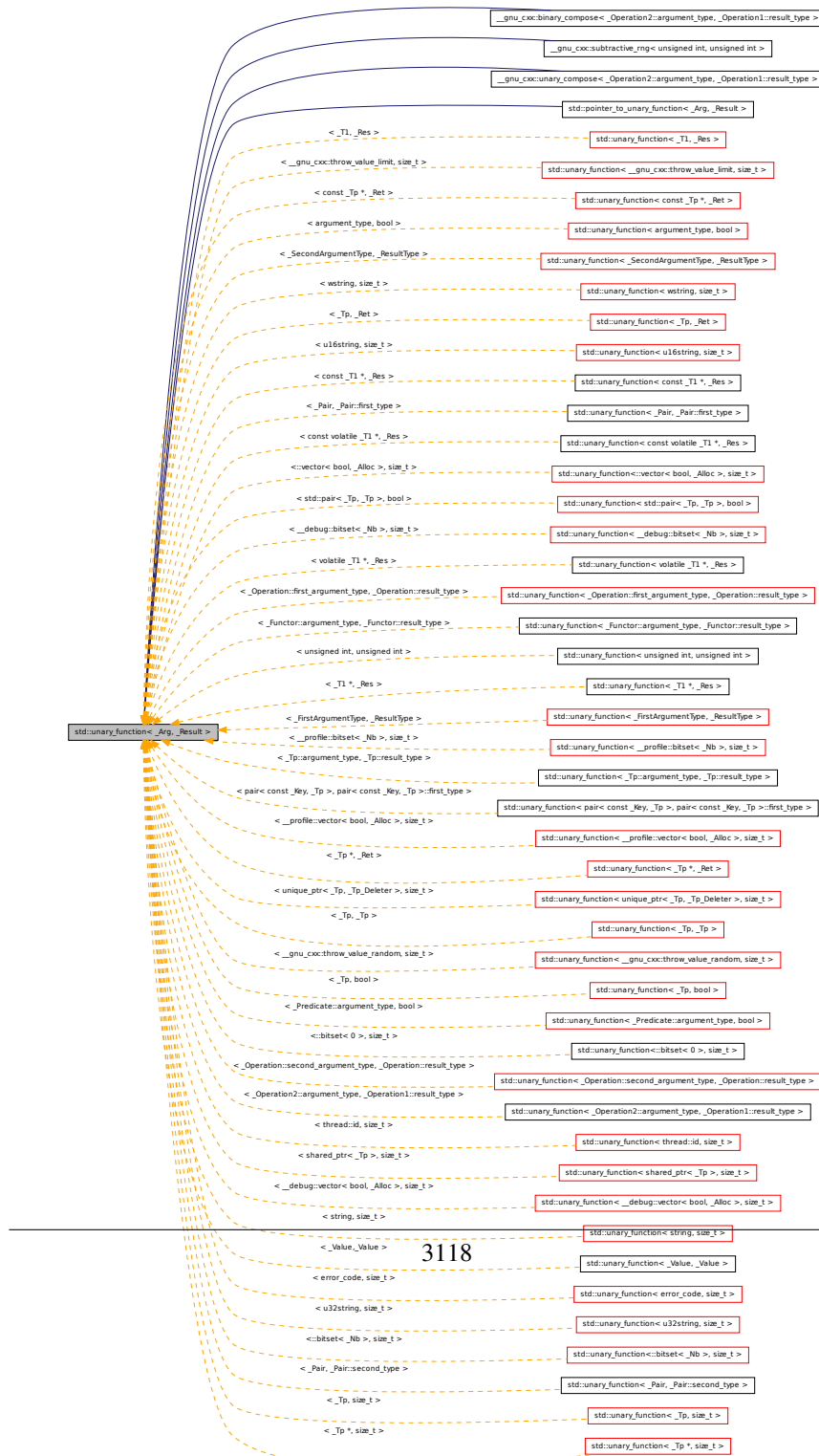
Definition at line 98 of file typeinfo.

The documentation for this class was generated from the following file:

- [typeinfo](#)

5.715 std::unary_function< _Arg, _Result > Struct Template Reference

Inheritance diagram for std::unary_function< _Arg, _Result >:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

5.715.1 Detailed Description

template<typename `_Arg`, typename `_Result`> struct `std::unary_function< _Arg, _Result >`

This is one of the [functor base classes](#).

Definition at line 100 of file `stl_function.h`.

5.715.2 Member Typedef Documentation

5.715.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.715.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

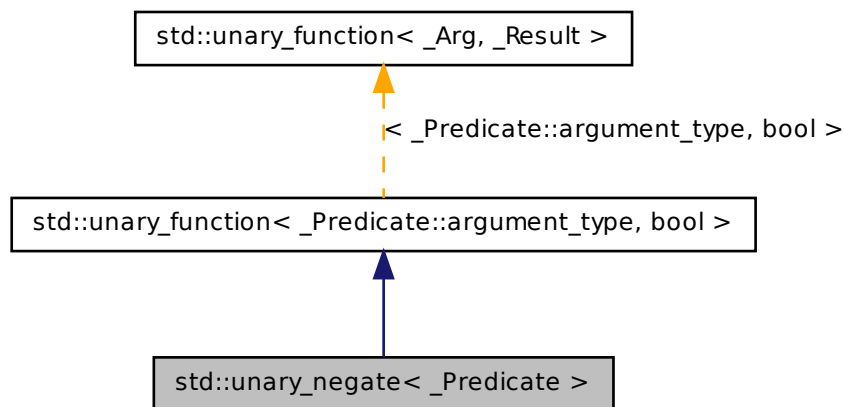
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.716 `std::unary_negate< _Predicate >` Class Template Reference

One of the [negation functors](#).

Inheritance diagram for `std::unary_negate<_Predicate>`:



Public Types

- typedef `_Predicate::argument_type` [argument_type](#)
- typedef `bool` [result_type](#)

Public Member Functions

- **`unary_negate`** (`const _Predicate &__x`)
- **`operator()`** (`const typename _Predicate::argument_type &__x`) `const`

Protected Attributes

- `_Predicate _M_pred`

5.716.1 Detailed Description

`template<typename _Predicate> class std::unary_negate<_Predicate>`

One of the [negation functors](#).

Definition at line 346 of file stl_function.h.

5.716.2 Member Typedef Documentation

5.716.2.1 `typedef _Predicate::argument_type std::unary_function<
_Predicate::argument_type , bool >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.716.2.2 `typedef bool std::unary_function< _Predicate::argument_type , bool
>::result_type [inherited]`

`result_type` is the return type

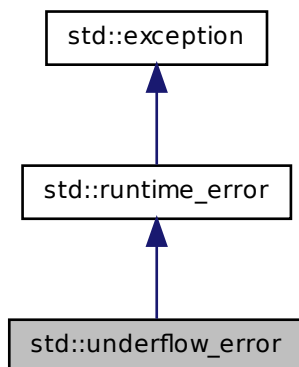
Definition at line 105 of file stl_function.h.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.717 std::underflow_error Class Reference

Inheritance diagram for std::underflow_error:



Public Member Functions

- **underflow_error** (const [string](#) &__arg)
- virtual const char * **what** () const throw ()

5.717.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 140 of file stdexcept.

5.717.2 Member Function Documentation

5.717.2.1 virtual const char* std::runtime_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

5.718 `std::uniform_int_distribution<_IntType>` Class Template Reference 3123

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.718 `std::uniform_int_distribution<_IntType>` Class Template Reference

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- [uniform_int_distribution](#) (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType>::max()`)
- [uniform_int_distribution](#) (const [param_type](#) &__p)
- [result_type a](#) () const
- [result_type b](#) () const
- [result_type max](#) () const
- [result_type min](#) () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param_type](#) &__param)
- [param_type param](#) () const
- void [reset](#) ()

Public Attributes

- [param_type _M_param](#)

5.718.1 Detailed Description

```
template<typename _IntType = int> class std::uniform_int_distribution< _-
_IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Definition at line 1628 of file random.h.

5.718.2 Member Typedef Documentation

5.718.2.1 `template<typename _IntType = int> typedef _IntType
std::uniform_int_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 1635 of file random.h.

5.718.3 Constructor & Destructor Documentation

5.718.3.1 `template<typename _IntType = int> std::uniform_int_distribution<
_IntType >::uniform_int_distribution (_IntType __a = 0, _IntType
__b = std::numeric_limits<_IntType>::max()) [inline,
explicit]`

Constructs a uniform distribution object.

Definition at line 1671 of file random.h.

5.718.4 Member Function Documentation

5.718.4.1 `template<typename _IntType = int> result_type
std::uniform_int_distribution< _IntType >::max () const
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1723 of file random.h.

5.718 std::uniform_int_distribution< _IntType > Class Template Reference 3125

5.718.4.2 `template<typename _IntType = int> result_type
std::uniform_int_distribution< _IntType >::min () const
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1716 of file random.h.

5.718.4.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::uniform_int_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 1731 of file random.h.

References `std::uniform_int_distribution< _IntType >::operator()()`, and
`std::uniform_int_distribution< _IntType >::param()`.

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`.

5.718.4.4 `template<typename _IntType = int> void std::uniform_int_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 1709 of file random.h.

5.718.4.5 `template<typename _IntType = int> param_type
std::uniform_int_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 1701 of file random.h.

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`,
`std::operator==()`, and `std::operator>>()`.

5.718.4.6 `template<typename _IntType = int> void
std::uniform_int_distribution< _IntType >::reset () [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1687 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.719 std::uniform_int_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef [uniform_int_distribution](#)< _IntType > **distribution_type**

Public Member Functions

- **param_type** ([_IntType](#) __a=0, [_IntType](#) __b=[std::numeric_limits](#)< [_IntType](#) >::max())
- [result_type](#) **a** () const
- [result_type](#) **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.719.1 Detailed Description

`template<typename _IntType = int> struct std::uniform_int_distribution< _IntType >::param_type`

Parameter type.

Definition at line 1637 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.720 `std::uniform_real_distribution<_RealType>` Class Template Reference

Uniform continuous distribution for random numbers.

Classes

- struct `param_type`

Public Types

- typedef `_RealType` `result_type`

Public Member Functions

- `uniform_real_distribution` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `uniform_real_distribution` (const `param_type` &__p)
- `result_type a` () const
- `result_type b` () const
- `result_type max` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator` >
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- void `param` (const `param_type` &__param)
- `param_type param` () const
- void `reset` ()

5.720.1 Detailed Description

```
template<typename _RealType = double> class std::uniform_real_distribution<  
_RealType >
```

Uniform continuous distribution for random numbers. A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1800 of file random.h.

5.720.2 Member Typedef Documentation

5.720.2.1 `template<typename _RealType = double> typedef _RealType
std::uniform_real_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 1807 of file random.h.

5.720.3 Constructor & Destructor Documentation

5.720.3.1 `template<typename _RealType = double> std::uniform_real_
distribution< _RealType >::uniform_real_distribution (_RealType
__a = _RealType(0), _RealType __b = _RealType(1))
[inline, explicit]`

Constructs a [uniform_real_distribution](#) object.

Parameters

`__min` [IN] The lower bound of the distribution.

`__max` [IN] The upper bound of the distribution.

Definition at line 1846 of file random.h.

5.720.4 Member Function Documentation

5.720.4.1 `template<typename _RealType = double> result_type
std::uniform_real_distribution< _RealType >::max () const
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1898 of file random.h.

5.720.4.2 `template<typename _RealType = double> result_type
std::uniform_real_distribution< _RealType >::min () const
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1891 of file random.h.

5.720 `std::uniform_real_distribution<_RealType>` Class Template Reference

5.720.4.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::uniform_real_distribution<_RealType>::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 1906 of file random.h.

References `std::uniform_real_distribution<_RealType>::operator()()`, and
`std::uniform_real_distribution<_RealType>::param()`.

Referenced by `std::uniform_real_distribution<_RealType>::operator()()`.

5.720.4.4 `template<typename _RealType = double> void
std::uniform_real_distribution<_RealType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 1884 of file random.h.

5.720.4.5 `template<typename _RealType = double> param_type
std::uniform_real_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 1876 of file random.h.

Referenced by `std::uniform_real_distribution<_RealType>::operator()()`,
`std::operator==()`, and `std::operator>>()`.

5.720.4.6 `template<typename _RealType = double> void
std::uniform_real_distribution<_RealType>::reset ()
[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1862 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

5.721 `std::uniform_real_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [uniform_real_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- [result_type](#) **a** () const
- [result_type](#) **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.721.1 Detailed Description

template<typename _RealType = double> struct std::uniform_real_distribution< _RealType >::param_type

Parameter type.

Definition at line 1809 of file [random.h](#).

The documentation for this struct was generated from the following file:

- [random.h](#)

5.722 `std::unique_lock< _Mutex >` Class Template Reference

[unique_lock](#)

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **unique_lock** (mutex_type &__m)
- **unique_lock** (mutex_type &__m, [try_to_lock_t](#))
- **unique_lock** (const [unique_lock](#) &)
- **unique_lock** (mutex_type &__m, [adopt_lock_t](#))
- **unique_lock** ([unique_lock](#) &&__u)
- **unique_lock** (mutex_type &__m, [defer_lock_t](#))
- template<typename _Clock , typename _Duration >
unique_lock (mutex_type &__m, const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- template<typename _Rep , typename _Period >
unique_lock (mutex_type &__m, const [chrono::duration](#)< _Rep, _Period > &__rtime)
- void **lock** ()
- mutex_type * **mutex** () const
- **operator bool** () const
- [unique_lock](#) & **operator=** (const [unique_lock](#) &)
- [unique_lock](#) & **operator=** ([unique_lock](#) &&__u)
- bool **owns_lock** () const
- mutex_type * **release** ()
- void **swap** ([unique_lock](#) &__u)
- bool **try_lock** ()
- template<typename _Rep , typename _Period >
bool **try_lock_for** (const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Clock , typename _Duration >
bool **try_lock_until** (const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- void **unlock** ()

5.722.1 Detailed Description

template<typename _Mutex> class std::unique_lock< _Mutex >

[unique_lock](#)

Definition at line 415 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

5.723 `std::unique_ptr< _Tp, _Tp_Deleter >` Class Template Reference

20.7.12.2 [unique_ptr](#) for single objects.

Public Types

- typedef `_Tp_Deleter deleter_type`
- typedef `_Tp element_type`
- typedef `_Pointer::type pointer`

Public Member Functions

- `unique_ptr` (pointer `__p`)
- `unique_ptr` (pointer `__p`, typename `std::remove_reference< deleter_type >::type &&__d`)
- `unique_ptr` (nullptr_t)
- `unique_ptr` (const `unique_ptr` &)
- `unique_ptr` (pointer `__p`, typename `std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d`)
- `unique_ptr` (`unique_ptr` &&__u)
- template<typename `_Up`, typename `_Up_Deleter` >
`unique_ptr` (`unique_ptr`< `_Up`, `_Up_Deleter` > &&__u)
- pointer `get` () const
- const deleter_type & `get_deleter` () const
- deleter_type & `get_deleter` ()
- `operator bool` () const
- `std::add_lvalue_reference< element_type >::type operator*` () const
- pointer `operator->` () const
- `unique_ptr` & `operator=` (const `unique_ptr` &)
- `unique_ptr` & `operator=` (`unique_ptr` &&__u)
- `unique_ptr` & `operator=` (nullptr_t)
- template<typename `_Up`, typename `_Up_Deleter` >
`unique_ptr` & `operator=` (`unique_ptr`< `_Up`, `_Up_Deleter` > &&__u)
- pointer `release` ()
- void `reset` (pointer `__p`=pointer())
- void `swap` (`unique_ptr` &__u)

5.723.1 Detailed Description

```
template<typename _Tp, typename _Tp_Deleter = default_delete<_Tp>> class
std::unique_ptr< _Tp, _Tp_Deleter >
```

20.7.12.2 [unique_ptr](#) for single objects.

Definition at line 81 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- [unique_ptr.h](#)

5.724 `std::unique_ptr< _Tp[], _Tp_Deleter >` Class Template Reference

20.7.12.3 [unique_ptr](#) for array objects with a runtime length

Public Types

- typedef `_Tp_Deleter deleter_type`
- typedef `_Tp element_type`
- typedef `_Tp * pointer`

Public Member Functions

- `unique_ptr` (pointer `__p`)
- template<typename `_Up` >
`unique_ptr` (`_Up *`, typename `std::remove_reference< deleter_type >::type` &&, typename `std::enable_if< std::is_convertible< _Up *, pointer >::value >::type` `*=0`)
- `unique_ptr` (pointer `__p`, typename `std::remove_reference< deleter_type >::type` && `__d`)
- `unique_ptr` (`nullptr_t`)
- `unique_ptr` (const `unique_ptr` &)
- `unique_ptr` (pointer `__p`, typename `std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type` `__d`)
- `unique_ptr` (`unique_ptr` && `__u`)
- template<typename `_Up` >
`unique_ptr` (`_Up *`, typename `std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type`, typename `std::enable_if< std::is_convertible< _Up *, pointer >::value >::type` `*=0`)

- `template<typename _Up >`
`unique_ptr` (`_Up *`, `typename std::enable_if< std::is_convertible< _Up *,`
`pointer >::value >::type !=0)`
- `template<typename _Up, typename _Up_Deleter >`
`unique_ptr` (`unique_ptr< _Up, _Up_Deleter > &&__u)`
- `pointer` `get` () `const`
- `deleter_type` & `get_deleter` ()
- `const deleter_type` & `get_deleter` () `const`
- `operator bool` () `const`
- `unique_ptr` & `operator=` (`nullptr_t`)
- `template<typename _Up, typename _Up_Deleter >`
`unique_ptr` & `operator=` (`unique_ptr< _Up, _Up_Deleter > &&__u)`
- `unique_ptr` & `operator=` (`unique_ptr` &&__u)
- `unique_ptr` & `operator=` (`const unique_ptr` &)
- `std::add_lvalue_reference< element_type >::type` `operator[]` (`size_t __i`) `const`
- `pointer` `release` ()
- `void` `reset` (`pointer __p=pointer()`)
- `void` `reset` (`nullptr_t`)
- `template<typename _Up >`
`void` `reset` (`_Up`)
- `void` `swap` (`unique_ptr` &__u)

5.724.1 Detailed Description

`template<typename _Tp, typename _Tp_Deleter> class std::unique_ptr< _Tp[],`
`_Tp_Deleter >`

20.7.12.3 `unique_ptr` for array objects with a runtime length

Definition at line 237 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- `unique_ptr.h`

5.725 `std::unordered_map< _Key, _Tp, _Hash, _Pred,` `_Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > **const_iterator**
- typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > **const_local_iterator**
- typedef _Allocator::const_pointer **const_pointer**
- typedef _Allocator::const_reference **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Base::hasher **hasher**
- typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Key **key_type**
- typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > **local_iterator**
- typedef _Allocator::pointer **pointer**
- typedef _Allocator::reference **reference**
- typedef _Base::size_type **size_type**
- typedef [_Base::value_type](#) **value_type**

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** ([unordered_map](#) &&__x)
- **unordered_map** ([initializer_list](#)< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const [unordered_map](#) &__x)
- const _RehashPolicy & **__rehash_policy** () const
- void **__rehash_policy** (const _RehashPolicy &)
- _Value_allocator_type **_M_get_Value_allocator** () const
- iterator **begin** ()
- const_iterator **begin** () const
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- size_type **bucket** (const key_type &__k) const

- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const
- const_local_iterator **cend** (size_type) const
- void **clear** ()
- size_type **count** (const key_type &__k) const
- bool **empty** () const
- local_iterator **end** (size_type)
- const_local_iterator **end** (size_type) const
- iterator **end** ()
- const_iterator **end** () const
- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__k)
- [std::pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- iterator **erase** (const_iterator, const_iterator)
- iterator **erase** (const_iterator)
- size_type **erase** (const key_type &)
- const_iterator **find** (const key_type &__k) const
- iterator **find** (const key_type &__k)
- allocator_type **get_allocator** () const
- [_Insert_Return_Type](#) **insert** (const value_type &__v)
- iterator **insert** (const_iterator, const value_type &__v)
- void **insert** ([initializer_list](#)< value_type > __l)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- key_equal **key_eq** () const
- float **load_factor** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- [unordered_map](#) & **operator=** (const [unordered_map](#) &__x)
- [unordered_map](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_map](#) & **operator=** ([unordered_map](#) &&__x)
- void **rehash** (size_type __n)
- size_type **size** () const
- void **swap** (_Hashtable &)

Friends

- struct [__detail::_Map_base](#)

5.725.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =  
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>  
>> class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys. Meets the requirements of a `container`, and `unordered associative container`

Parameters

Key Type of key objects.

Tp Type of mapped objects.

Hash Hashing function object type, defaults to `hash<Value>`.

Pred Predicate function object type, defaults to `equal_to<Value>`.

Alloc Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 224 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered_map.h](#)

5.726 `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::__Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::__Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`

- typedef std::ptrdiff_t **difference_type**
- typedef _Base::hasher **hasher**
- typedef __detail::Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Key **key_type**
- typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > **local_iterator**
- typedef _Allocator::pointer **pointer**
- typedef _Allocator::reference **reference**
- typedef _Base::size_type **size_type**
- typedef [_Base::value_type](#) **value_type**

Public Member Functions

- **unordered_multimap** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multimap (_InputIterator __f, _InputIterator __l, typename _Base::size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** ([unordered_multimap](#) &&__x)
- **unordered_multimap** ([initializer_list](#)< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (const [unordered_multimap](#) &__x)
- const _RehashPolicy & **__rehash_policy** () const
- void **__rehash_policy** (const _RehashPolicy &)
- _Value_allocator_type **_M_get_Value_allocator** () const
- iterator **begin** ()
- const_iterator **begin** () const
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const
- const_local_iterator **cend** (size_type) const
- void **clear** ()
- size_type **count** (const key_type &__k) const
- bool **empty** () const

- `local_iterator end` (`size_type`)
- `const_local_iterator end` (`size_type`) `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `std::pair< iterator, iterator > equal_range` (`const key_type &__k`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__k`) `const`
- `iterator erase` (`const_iterator, const_iterator`)
- `iterator erase` (`const_iterator`)
- `size_type erase` (`const key_type &`)
- `const_iterator find` (`const key_type &__k`) `const`
- `iterator find` (`const key_type &__k`)
- `allocator_type get_allocator` () `const`
- `_Insert_Return_Type insert` (`const value_type &__v`)
- `iterator insert` (`const_iterator, const value_type &__v`)
- `void insert` (`initializer_list< value_type > __l`)
- `template<typename _InputIterator >`
`void insert` (`_InputIterator __first, _InputIterator __last`)
- `key_equal key_eq` () `const`
- `float load_factor` () `const`
- `size_type max_bucket_count` () `const`
- `size_type max_size` () `const`
- `unordered_multimap & operator=` (`const unordered_multimap &__x`)
- `unordered_multimap & operator=` (`initializer_list< value_type > __l`)
- `unordered_multimap & operator=` (`unordered_multimap &&__x`)
- `void rehash` (`size_type __n`)
- `size_type size` () `const`
- `void swap` (`_Hashtable &`)

Friends

- `struct __detail::Map_base`

5.726.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys. Meets the requirements of a `container`, and `unordered associative container`

Parameters

Key Type of key objects.

Tp Type of mapped objects.

Hash Hashing function object type, defaults to `hash<Value>`.

Pred Predicate function object type, defaults to `equal_to<Value>`.

Alloc Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 310 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered_map.h](#)

5.727 `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Inherits `std::__unordered_multiset<_Value, _Hash, _Pred, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multiset (_InputIterator __f, _InputIterator __l, typename _Base::size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (unordered_multiset &&__x)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const unordered_multiset &__x)
- const _RehashPolicy & __**rehash_policy** () const
- void __**rehash_policy** (const _RehashPolicy &)
- _Value_allocator_type **M_get_Value_allocator** () const
- iterator **begin** ()
- const_iterator **begin** () const
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const
- const_local_iterator **cend** (size_type) const
- void **clear** ()
- size_type **count** (const key_type &__k) const
- bool **empty** () const
- local_iterator **end** (size_type)
- const_local_iterator **end** (size_type) const
- iterator **end** ()
- const_iterator **end** () const
- std::pair< iterator, iterator > **equal_range** (const key_type &__k)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- iterator **erase** (const_iterator, const_iterator)
- iterator **erase** (const_iterator)
- size_type **erase** (const key_type &)
- const_iterator **find** (const key_type &__k) const
- iterator **find** (const key_type &__k)
- allocator_type **get_allocator** () const

- [_Insert_Return_Type](#) **insert** (const value_type &__v)
- iterator **insert** (const_iterator, const value_type &__v)
- void **insert** ([initializer_list](#)< value_type > __l)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- key_equal **key_eq** () const
- float **load_factor** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- [unordered_multiset](#) & **operator=** (const [unordered_multiset](#) &__x)
- [unordered_multiset](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_multiset](#) & **operator=** ([unordered_multiset](#) &&__x)
- void **rehash** (size_type __n)
- size_type **size** () const
- void **swap** (_Hashtable &)

Friends

- struct [__detail::_Map_base](#)

5.727.1 Detailed Description

template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

Parameters

Value Type of key objects.

Hash Hashing function object type, defaults to hash<Value>.

Pred Predicate function object type, defaults to equal_to<Value>.

Alloc Allocator type, defaults to allocator<Key>.

Definition at line 300 of file unordered_set.h.

The documentation for this class was generated from the following file:

- [unordered_set.h](#)

5.728 `std::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Inherits `std::__unordered_set< _Value, _Hash, _Pred, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::__Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::__Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::__Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::__Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- `unordered_set (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator >
unordered_set (_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_set (unordered_set &&__x)`
- `unordered_set (initializer_list< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_set (const unordered_set &__x)`

- `const _RehashPolicy & __rehash_policy () const`
- `void __rehash_policy (const _RehashPolicy &)`
- `_Value_allocator_type _M_get_Value_allocator () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `local_iterator begin (size_type __n)`
- `const_local_iterator begin (size_type __n) const`
- `size_type bucket (const key_type &__k) const`
- `size_type bucket_count () const`
- `size_type bucket_size (size_type __n) const`
- `const_iterator cbegin () const`
- `const_local_iterator cbegin (size_type __n) const`
- `const_iterator cend () const`
- `const_local_iterator cend (size_type) const`
- `void clear ()`
- `size_type count (const key_type &__k) const`
- `bool empty () const`
- `local_iterator end (size_type)`
- `const_local_iterator end (size_type) const`
- `iterator end ()`
- `const_iterator end () const`
- `std::pair< iterator, iterator > equal_range (const key_type &__k)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__k) const`
- `iterator erase (const_iterator, const_iterator)`
- `iterator erase (const_iterator)`
- `size_type erase (const key_type &)`
- `const_iterator find (const key_type &__k) const`
- `iterator find (const key_type &__k)`
- `allocator_type get_allocator () const`
- `_Insert_Return_Type insert (const value_type &__v)`
- `iterator insert (const_iterator, const value_type &__v)`
- `void insert (initializer_list< value_type > __l)`
- `template<typename _InputIterator >
void insert (_InputIterator __first, _InputIterator __last)`
- `key_equal key_eq () const`
- `float load_factor () const`
- `size_type max_bucket_count () const`
- `size_type max_size () const`
- `unordered_set & operator= (const unordered_set &__x)`
- `unordered_set & operator= (initializer_list< value_type > __l)`
- `unordered_set & operator= (unordered_set &&__x)`
- `void rehash (size_type __n)`
- `size_type size () const`
- `void swap (_Hashtable &)`

Friends

- `struct __detail::_Map_base`

5.728.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_set<_Value, _Hash, _Pred, _Alloc >`

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

Parameters

Value Type of key objects.

Hash Hashing function object type, defaults to `hash<Value>`.

Pred Predicate function object type, defaults to `equal_to<Value>`.

Alloc Allocator type, defaults to `allocator<Key>`.

Definition at line 217 of file `unordered_set.h`.

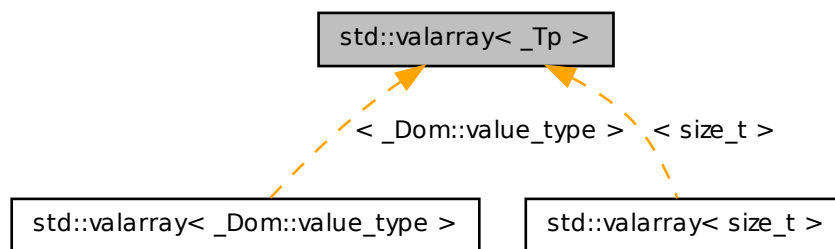
The documentation for this class was generated from the following file:

- [unordered_set.h](#)

5.729 `std::valarray<_Tp>` Class Template Reference

Smart array designed to support numeric processing.

Inheritance diagram for `std::valarray< _Tp >`:



Public Types

- `typedef _Tp value_type`

Public Member Functions

- `valarray ()`
- `valarray (size_t)`
- `valarray (const _Tp *__restrict__, size_t)`
- `valarray (const mask_array< _Tp > &)`
- `valarray (const indirect_array< _Tp > &)`
- `template<typename _Tp>`
`valarray (const _Tp *__restrict__ __p, size_t __n)`
- `valarray (const valarray &)`
- `valarray (initializer_list< _Tp >)`
- `template<class _Dom >`
`valarray (const _Expr< _Dom, _Tp > &__e)`
- `valarray (const _Tp &, size_t)`
- `valarray (const slice_array< _Tp > &)`
- `valarray (const gslice_array< _Tp > &)`
- `_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > apply (_Tp func(_Tp)) const`
- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > apply (_Tp func(const _Tp &)) const`
- `valarray< _Tp > cshift (int) const`
- `_Tp max () const`

- `_Tp min () const`
- `_UnaryOp< __logical_not >::_Rt operator! () const`
- `valarray<_Tp> & operator%= (const _Tp &)`
- `valarray<_Tp> & operator%= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator%= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator&= (const _Tp &)`
- `valarray<_Tp> & operator&= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator&= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator*= (const _Tp &)`
- `valarray<_Tp> & operator*= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator*= (const _Expr<_Dom, _Tp> &)`
- `_UnaryOp< __unary_plus >::_Rt operator+ () const`
- `valarray<_Tp> & operator+= (const _Tp &)`
- `valarray<_Tp> & operator+= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator+= (const _Expr<_Dom, _Tp> &)`
- `_UnaryOp< __negate >::_Rt operator- () const`
- `valarray<_Tp> & operator-= (const _Tp &)`
- `valarray<_Tp> & operator-= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator-= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator/= (const _Tp &)`
- `valarray<_Tp> & operator/= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator/= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator<<= (const _Tp &)`
- `valarray<_Tp> & operator<<= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & operator<<= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator= (const _Tp &)`
- `valarray<_Tp> & operator= (const gslice_array<_Tp> &)`
- `valarray<_Tp> & operator= (const slice_array<_Tp> &)`
- `valarray<_Tp> & operator= (const indirect_array<_Tp> &)`
- `valarray & operator= (initializer_list<_Tp>)`
- `template<class _Dom>`
`valarray<_Tp> & operator= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator= (const mask_array<_Tp> &)`
- `valarray<_Tp> & operator= (const valarray<_Tp> &)`
- `valarray<_Tp> & operator>>= (const valarray<_Tp> &)`

- `template<class _Dom >`
`valarray< _Tp > & operator>= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator>= (const _Tp &)`
- `gslice_array< _Tp > operator[] (const gslice &)`
- `_Tp & operator[] (size_t)`
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > operator[] (slice) const`
- `valarray< _Tp > operator[] (const valarray< bool > &) const`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > operator[] (const valarray< size_t > &) const`
- `slice_array< _Tp > operator[] (slice)`
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > operator[] (const gslice &) const`
- `indirect_array< _Tp > operator[] (const valarray< size_t > &)`
- `const _Tp & operator[] (size_t) const`
- `mask_array< _Tp > operator[] (const valarray< bool > &)`
- `valarray< _Tp > & operator^= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & operator^= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator^= (const _Tp &)`
- `template<class _Dom >`
`valarray< _Tp > & operator|= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator|= (const _Tp &)`
- `valarray< _Tp > & operator|= (const valarray< _Tp > &)`
- `_UnaryOp< __bitwise_not >::_Rt operator~ () const`
- `void resize (size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > shift (int) const`
- `size_t size () const`
- `_Tp sum () const`

Friends

- `class _Array< _Tp >`

5.729.1 Detailed Description

`template<class _Tp> class std::valarray< _Tp >`

Smart array designed to support numeric processing. A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

Tp Type of object in the array.

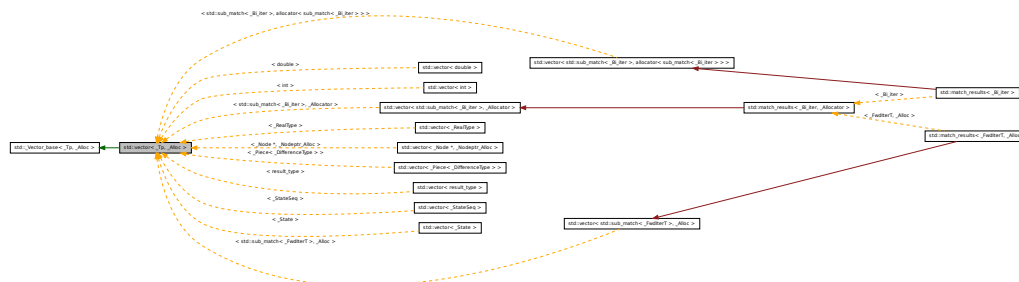
5.729.2 Constructor & Destructor Documentation

Construct an array initialized to the first n elements of t .

- `valarray`

A standard container which offers fixed time access to individual elements in any order.

Inheritance diagram for `std::vector<_Tp, _Alloc>`:



- typedef `_Alloc` **allocator_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**

- typedef [std::reverse_iterator](#)< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, [vector](#) > **iterator**
- typedef _Tp_alloc_type::pointer **pointer**
- typedef _Tp_alloc_type::reference **reference**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [vector](#) ()
- [vector](#) (const allocator_type &__a)
- [vector](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
[vector](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- [vector](#) (const [vector](#) &__x)
- [vector](#) (size_type __n)
- [vector](#) ([vector](#) &&__x)
- [vector](#) ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- [~vector](#) ()
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename _InputIterator >
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) ([initializer_list](#)< value_type > __l)
- reference [at](#) (size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- size_type [capacity](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- [const_reverse_iterator](#) [crbegin](#) () const
- [const_reverse_iterator](#) [crend](#) () const
- _Tp * [data](#) ()
- const _Tp * [data](#) () const

- template<typename... _Args>
iterator **emplace** (iterator __position, _Args &&...__args)
- template<typename... _Args>
void **emplace_back** (_Args &&...__args)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- const_reference **front** () const
- reference **front** ()
- iterator **insert** (iterator __position, value_type &&__x)
- template<typename _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- void **insert** (iterator __position, **initializer_list**< value_type > __l)
- void **insert** (iterator __position, size_type __n, const value_type &__x)
- iterator **insert** (iterator __position, const value_type &__x)
- size_type **max_size** () const
- **vector** & **operator=** (**initializer_list**< value_type > __l)
- **vector** & **operator=** (**vector** &&__x)
- **vector** & **operator=** (const **vector** &__x)
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- void **pop_back** ()
- void **push_back** (value_type &&__x)
- void **push_back** (const value_type &__x)
- **reverse_iterator** **rbegin** ()
- **const_reverse_iterator** **rbegin** () const
- **const_reverse_iterator** **rend** () const
- **reverse_iterator** **rend** ()
- void **reserve** (size_type __n)
- void **resize** (size_type __new_size)
- void **resize** (size_type __new_size, const value_type &__x)
- void **shrink_to_fit** ()
- size_type **size** () const
- void **swap** (**vector** &__x)

Protected Member Functions

- `_Tp_alloc_type::pointer M_allocate (size_t __n)`
- `template<typename _ForwardIterator >
pointer M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator >
void M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >
void M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _Integer >
void M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >
void M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type M_check_len (size_type __n, const char *__s) const`
- `void M_deallocate (typename _Tp_alloc_type::pointer __p, size_t __n)`
- `void M_default_append (size_type __n)`
- `void M_default_initialize (size_type __n)`
- `void M_erase_at_end (pointer __pos)`
- `void M_fill_assign (size_type __n, const value_type &__val)`
- `void M_fill_initialize (size_type __n, const value_type &__value)`
- `void M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Tp_alloc_type & M_get_Tp_allocator ()`
- `const _Tp_alloc_type & M_get_Tp_allocator () const`
- `template<typename _InputIterator >
void M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _Integer >
void M_initialize_dispatch (_Integer __n, _Integer __value, __true_type)`
- `template<typename... _Args>
void M_insert_aux (iterator __position, _Args &&...__args)`
- `template<typename _InputIterator >
void M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _Integer >
void M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `void M_range_check (size_type __n) const`
- `template<typename _ForwardIterator >
void M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`

- `template<typename _InputIterator >`
`void _M_range_initialize` (`_InputIterator __first`, `_InputIterator __last`,
[std::input_iterator_tag](#))
- `template<typename _ForwardIterator >`
`void _M_range_insert` (`iterator __pos`, `_ForwardIterator __first`, `_-`
`ForwardIterator __last`, [std::forward_iterator_tag](#))
- `template<typename _InputIterator >`
`void _M_range_insert` (`iterator __pos`, `_InputIterator __first`, `_InputIterator __-`
`last`, [std::input_iterator_tag](#))
- `allocator_type` `get_allocator` () `const`

Protected Attributes

- `_Vector_impl _M_impl`

5.730.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class`
`std::vector<_Tp, _Alloc>`

A standard container which offers fixed time access to individual elements in any order. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting (`[]`) access is also provided as with C-style arrays.

Definition at line 178 of file `std_vector.h`.

5.730.2 Constructor & Destructor Documentation

5.730.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`
`std::vector<_Tp, _Alloc>::vector ()` [`inline`]

Default constructor creates no elements.

Definition at line 215 of file `std_vector.h`.

5.730.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (const allocator_type & __a)
[inline, explicit]`

Creates a vector with no elements.

Parameters

a An allocator object.

Definition at line 223 of file `stl_vector.h`.

5.730.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (size_type __n) [inline,
explicit]`

Creates a vector with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor fills the vector with *n* default constructed elements.

Definition at line 235 of file `stl_vector.h`.

5.730.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (size_type __n, const value_type
& __value, const allocator_type & __a = allocator_type())
[inline]`

Creates a vector with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

a An allocator.

This constructor fills the vector with *n* copies of *value*.

Definition at line 247 of file `stl_vector.h`.

5.730.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (const vector< _Tp, _Alloc > &
__x) [inline]`

Vector copy constructor.

Parameters

x A vector of identical element and allocator types.

The newly-created vector uses a copy of the allocation object used by *x*. All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied.

Definition at line 276 of file stl_vector.h.

5.730.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (vector< _Tp, _Alloc > && __x
) [inline]`

Vector move constructor.

Parameters

x A vector of identical element and allocator types.

The newly-created vector contains the exact contents of *x*. The contents of *x* are a valid, but unspecified vector.

Definition at line 292 of file stl_vector.h.

5.730.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (initializer_list< value_type
> __l, const allocator_type & __a = allocator_type())
[inline]`

Builds a vector from an initializer list.

Parameters

l An [initializer_list](#).

a An allocator.

Create a vector consisting of copies of the elements in the [initializer_list](#) *l*.

This will call the element type's copy constructor *N* times (where *N* is *l.size()*) and do no memory reallocation.

Definition at line 306 of file stl_vector.h.

```
5.730.2.8  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             template<typename _InputIterator > std::vector< _Tp, _Alloc
             >::vector ( _InputIterator __first, _InputIterator __last, const
             allocator_type & __a = allocator_type() ) [inline]
```

Builds a vector from a range.

Parameters

first An input iterator.

last An input iterator.

a An allocator.

Create a vector consisting of copies of the elements from [first,last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 332 of file stl_vector.h.

```
5.730.2.9  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             std::vector< _Tp, _Alloc >::~~vector ( ) [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 347 of file stl_vector.h.

5.730.3 Member Function Documentation

```
5.730.3.1  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             template<typename _ForwardIterator > pointer std::vector< _Tp,
             _Alloc >::_M_allocate_and_copy ( size_type __n, _ForwardIterator
             __first, _ForwardIterator __last ) [inline, protected]
```

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

Definition at line 1047 of file stl_vector.h.

Referenced by std::vector< _Tp, _Alloc >::operator=().

5.730.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::M_range_check (size_type __n)
const [inline, protected]`

Safety check used only from [at\(\)](#).

Definition at line 714 of file `stl_vector.h`.

5.730.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::vector< _Tp,
_Alloc >::assign (_InputIterator __first, _InputIterator __last)
[inline]`

Assigns a range to a vector.

Parameters

first An input iterator.

last An input iterator.

This function fills a vector with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 427 of file `stl_vector.h`.

5.730.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::assign (size_type __n, const
value_type & __val) [inline]`

Assigns a given value to a vector.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a vector with *n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 410 of file `stl_vector.h`.

```
5.730.3.5  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::assign ( initializer_list< value_type
             > __l ) [inline]
```

Assigns an initializer list to a vector.

Parameters

l An [initializer_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 447 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::assign()`.

```
5.730.3.6  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             reference std::vector< _Tp, _Alloc >::at ( size_type __n )
             [inline]
```

Provides access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws [out_of_range](#) if the check fails.

Definition at line 733 of file `stl_vector.h`.

```
5.730.3.7  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             const_reference std::vector< _Tp, _Alloc >::at ( size_type __n )
             const [inline]
```

Provides access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

[`std::out_of_range`](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws [`out_of_range`](#) if the check fails.

Definition at line 751 of file `stl_vector.h`.

5.730.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::back () const
[inline]`

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 786 of file `stl_vector.h`.

5.730.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::back () [inline]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 778 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::max()`, and `std::piecewise_constant_distribution<_RealType>::max()`.

5.730.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::begin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Reimplemented in [`std::match_results<_Bi_iter, _Allocator>`](#), [`std::match_results<_Bi_iter>`](#), and [`std::match_results<_FwdIterT, _Alloc>`](#).

Definition at line 470 of file `stl_vector.h`.

5.730.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::begin () [inline]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 461 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc>::emplace()`, `std::vector<_Tp, _Alloc>::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector<_Tp, _Alloc>::operator=()`, `std::vector::operator==()`, and `std::vector< std::sub_match<_Bi_iter>, _Allocator>::vector()`.

5.730.3.12 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector<_Tp, _Alloc>::capacity () const
[inline]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 648 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc>::operator=()`.

5.730.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::cbegin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 534 of file `stl_vector.h`.

5.730.3.14 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::cend () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 543 of file `stl_vector.h`.

5.730.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::clear () [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1037 of file stl_vector.h.

5.730.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin ()
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 552 of file stl_vector.h.

5.730.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 561 of file stl_vector.h.

5.730.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp* std::vector< _Tp, _Alloc >::data () [inline]`

Returns a pointer such that [`data()`, `data() + size()`) is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 801 of file stl_vector.h.

5.730.3.19 `template<typename _Tp , typename _Alloc > template<typename...
_Args> vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc
>::emplace (iterator __position, _Args &&... __args)`

Inserts an object in vector before specified iterator.

Parameters

position An iterator into the vector.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.`

Definition at line 272 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

5.730.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector<_Tp, _Alloc>::empty () const [inline]`

Returns true if the vector is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _-Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 657 of file `stl_vector.h`.

5.730.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::end () [inline]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 479 of file `stl_vector.h`.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::vector< _Tp, _-Alloc >::erase()`, `std::vector< _Tp, _Alloc >::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::vector()`.

5.730.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::end () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _-Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 488 of file `stl_vector.h`.

5.730.3.23 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::erase (iterator __position)`

Remove element at given position.

Parameters

position Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 134 of file vector.tcc.

References `std::vector< _Tp, _Alloc >::end()`.

5.730.3.24 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::erase (iterator __first, iterator __last)`

Remove a range of elements.

Parameters

first Iterator pointing to the first element to be erased.

last Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range [first,last) and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 146 of file vector.tcc.

References `std::vector< _Tp, _Alloc >::end()`.

5.730.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::front () [inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 762 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::min()`, and `std::piecewise_constant_distribution<_RealType>::min()`.

5.730.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 770 of file `stl_vector.h`.

5.730.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator> void std::vector<_Tp,
_Alloc>::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]`

Inserts a range into the vector.

Parameters

position An iterator into the vector.

first An input iterator.

last An input iterator.

This function will insert copies of the data in the range `[first,last)` into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 960 of file `stl_vector.h`.

5.730.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::insert (iterator __position,
value_type && __x) [inline]`

Inserts given rvalue into vector before specified iterator.

Parameters

position An iterator into the vector.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 906 of file stl_vector.h.

```
5.730.3.29  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::insert ( iterator __position,
             initializer_list< value_type > __l ) [inline]
```

Inserts an [initializer_list](#) into the vector.

Parameters

position An iterator into the vector.

l An [initializer_list](#).

This function will insert copies of the data in the [initializer_list](#) *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 923 of file stl_vector.h.

Referenced by std::vector< std::sub_match< _Bi_iter >, _Allocator >::insert().

```
5.730.3.30  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::insert ( iterator __position,
             size_type __n, const value_type & __x ) [inline]
```

Inserts a number of copies of given data into the vector.

Parameters

position An iterator into the vector.

n Number of elements to be inserted.

x Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 941 of file `stl_vector.h`.

5.730.3.31 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::insert (iterator __position, const value_type & __x)`

Inserts given value into vector before specified iterator.

Parameters

position An iterator into the vector.
x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 107 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

5.730.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector< _Tp, _Alloc >::max_size () const [inline]`

Returns the [size\(\)](#) of the largest possible vector.

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 573 of file `stl_vector.h`.

5.730.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector< _Tp, _Alloc >::operator= (vector< _Tp, _Alloc > && __x) [inline]`

Vector move assignment operator.

Parameters

x A vector of identical element and allocator types.

The contents of *x* are moved into this vector (without copying). *x* is a valid, but unspecified vector.

Definition at line 371 of file stl_vector.h.

5.730.3.34 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc > & std::vector< _Tp, _Alloc >::operator= (const vector< _Tp, _Alloc > & __x)`

Vector assignment operator.

Parameters

x A vector of identical element and allocator types.

All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

Definition at line 157 of file vector.tcc.

References std::_Destroy(), std::vector< _Tp, _Alloc >::_M_allocate_and_copy(), std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::capacity(), std::vector< _Tp, _Alloc >::end(), and std::vector< _Tp, _Alloc >::size().

5.730.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector< _Tp, _Alloc >::operator= (initializer_list< value_type > __l) [inline]`

Vector list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 392 of file stl_vector.h.

5.730.3.36 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::operator[] (size_type
__n) const [inline]`

Subscript access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 708 of file `stl_vector.h`.

5.730.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (size_type __n)
[inline]`

Subscript access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 693 of file `stl_vector.h`.

5.730.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back () [inline]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before [pop_back\(\)](#) is called.

Definition at line 855 of file stl_vector.h.

5.730.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::push_back (const value_type &
__x) [inline]`

Add data to the end of the vector.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 824 of file stl_vector.h.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

5.730.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector< _Tp, _Alloc >::rbegin ()
[inline]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 497 of file stl_vector.h.

5.730.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 506 of file stl_vector.h.

5.730.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector< _Tp, _Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 515 of file stl_vector.h.

5.730.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 524 of file `stl_vector.h`.

5.730.3.44 `template<typename _Tp , typename _Alloc > void std::vector<
_Tp, _Alloc >::reserve (size_type __n)`

Attempt to preallocate enough memory for specified number of elements.

Parameters

n Number of elements required.

Exceptions

std::length_error If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 65 of file `vector.tcc`.

References `std::_Destroy()`.

5.730.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (size_type __new_size,
const value_type & __x) [inline]`

Resizes the vector to the specified number of elements.

Parameters

new_size Number of elements the vector should contain.

x Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 607 of file stl_vector.h.

```
5.730.3.46  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::resize ( size_type __new_size )
             [inline]
```

Resizes the vector to the specified number of elements.

Parameters

new_size Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 587 of file stl_vector.h.

Referenced by __gnu_parallel::__shrink_and_double(), __gnu_parallel::multiway_merge_exact_splitting(), and __gnu_parallel::parallel_sort_mwms().

```
5.730.3.47  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::shrink_to_fit ( ) [inline]
```

A non-binding request to reduce [capacity\(\)](#) to [size\(\)](#).

Definition at line 639 of file stl_vector.h.

```
5.730.3.48  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             size_type std::vector< _Tp, _Alloc >::size ( ) const [inline]
```

Returns the number of elements in the vector.

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _-Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 568 of file stl_vector.h.

Referenced by __gnu_parallel::__shrink(), __gnu_parallel::__shrink_and_double(), __gnu_parallel::list_partition(), std::discrete_distribution< _IntType >::max(), std::vector< _Tp, _Alloc >::operator=(), and std::operator==().

5.730.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::vector<_Tp, _Alloc>::swap (vector<_Tp, _Alloc> &
 __x) [inline]`

Swaps data with another vector.

Parameters

x A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 1017 of file `stl_vector.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following files:

- [stl_vector.h](#)
- [vector.tcc](#)

5.731 `std::vector< bool, _Alloc >` Class Template Reference

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Inherits `std::_Bvector_base< _Alloc >`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Bit_const_iterator` **const_iterator**
- typedef `const bool *` **const_pointer**
- typedef `bool` **const_reference**
- typedef [std::reverse_iterator](#)< `const_iterator` > **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef [std::reverse_iterator](#)< `iterator` > **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `bool` **value_type**

Public Member Functions

- **vector** (size_type __n, const bool &__value=bool(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- **vector** (const [vector](#) &__x)
- **vector** (const allocator_type &__a)
- **vector** ([vector](#) &&__x)
- **vector** ([initializer_list](#)< bool > __l, const allocator_type &__a=allocator_type())
- void **assign** (size_type __n, const bool &__x)
- template<typename _InputIterator >
void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** ([initializer_list](#)< bool > __l)
- reference **at** (size_type __n)
- const_reference **at** (size_type __n) const
- reference **back** ()
- const_reference **back** () const
- const_iterator **begin** () const
- iterator **begin** ()
- size_type **capacity** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- void **data** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- void **flip** ()
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- void **insert** (iterator __position, size_type __n, const bool &__x)
- iterator **insert** (iterator __position, const bool &__x=bool())
- template<typename _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- void **insert** (iterator __p, [initializer_list](#)< bool > __l)

- `size_type max_size () const`
- `vector & operator= (vector &&__x)`
- `vector & operator= (const vector &__x)`
- `vector & operator= (initializer_list< bool > __l)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n) const`
- `void pop_back ()`
- `void push_back (bool __x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void reserve (size_type __n)`
- `void resize (size_type __new_size, bool __x=bool())`
- `void shrink_to_fit ()`
- `size_type size () const`
- `void swap (vector &__x)`

Static Public Member Functions

- `static void swap (reference __x, reference __y)`

Protected Types

- `typedef _Alloc::template rebind< _Bit_type >::other _Bit_alloc_type`

Protected Member Functions

- `_Bit_type * _M_allocate (size_t __n)`
- `template<typename _InputIterator >`
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _Integer >`
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<class _InputIterator >`
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`

- iterator **_M_copy_aligned** (const_iterator __first, const_iterator __last, iterator __result)
- void **_M_deallocate** ()
- void **_M_erase_at_end** (iterator __pos)
- void **_M_fill_assign** (size_t __n, bool __x)
- void **_M_fill_insert** (iterator __position, size_type __n, bool __x)
- `_Bit_alloc_type` & **_M_get_Bit_allocator** ()
- const `_Bit_alloc_type` & **_M_get_Bit_allocator** () const
- void **_M_initialize** (size_type __n)
- template<typename _InputIterator >
void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer >
void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
void **_M_initialize_range** (_InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- template<typename _ForwardIterator >
void **_M_initialize_range** (_ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- void **_M_insert_aux** (iterator __position, bool __x)
- template<typename _Integer >
void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- template<typename _ForwardIterator >
void **_M_insert_range** (iterator __position, _ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- template<typename _InputIterator >
void **_M_insert_range** (iterator __pos, _InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- void **_M_range_check** (size_type __n) const

Protected Attributes

- `_Bvector_impl` **_M_impl**

Friends

- class **hash**

5.731.1 Detailed Description

template<typename _Alloc> class std::vector< bool, _Alloc >

A specialization of vector for booleans which offers fixed time access to individual elements in any order. Note that vector<bool> does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to bool. See DR96 for details.

See also

[vector](#) for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

Definition at line 474 of file stl_bvector.h.

The documentation for this class was generated from the following files:

- [stl_bvector.h](#)
- [vector.tcc](#)

5.732 std::weak_ptr< _Tp > Class Template Reference

A smart pointer with weak semantics.

Inherits `__weak_ptr< _Tp >`.

Public Member Functions

- `template<typename _Tp1 >`
weak_ptr (const [weak_ptr](#)< _Tp1 > &__r)
- `template<typename _Tp1 >`
weak_ptr (const [shared_ptr](#)< _Tp1 > &__r)
- [shared_ptr](#)< _Tp > **lock** () const
- `template<typename _Tp1 >`
[weak_ptr](#) & **operator=** (const [shared_ptr](#)< _Tp1 > &__r)
- `template<typename _Tp1 >`
[weak_ptr](#) & **operator=** (const [weak_ptr](#)< _Tp1 > &__r)

5.732.1 Detailed Description

`template<typename _Tp> class std::weak_ptr<_Tp>`

A smart pointer with weak semantics. With forwarding constructors and assignment operators.

Definition at line 364 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

5.733 `std::weibull_distribution<_RealType>` Class Template Reference

A [weibull_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **`weibull_distribution`** (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- **`weibull_distribution`** (`const param_type &__p`)
- `_RealType a` () const
- `_RealType b` () const
- [result_type max](#) () const
- [result_type min](#) () const
- `template<typename _UniformRandomNumberGenerator>`
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator>`
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (`const param_type &__param`)
- [param_type param](#) () const
- void [reset](#) ()

5.733.1 Detailed Description

template<typename _RealType = double> class std::weibull_distribution< _RealType >

A [weibull_distribution](#) random number distribution. The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$$

Definition at line 4335 of file random.h.

5.733.2 Member Typedef Documentation

5.733.2.1 template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 4342 of file random.h.

5.733.3 Member Function Documentation

5.733.3.1 template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::a () const [inline]

Return the a parameter of the distribution.

Definition at line 4393 of file random.h.

5.733.3.2 template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::b () const [inline]

Return the b parameter of the distribution.

Definition at line 4400 of file random.h.

5.733.3.3 template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::max () const [inline]

Returns the least upper bound value of the distribution.

Definition at line 4429 of file random.h.

5.733.3.4 `template<typename _RealType = double> result_type
std::weibull_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4422 of file random.h.

5.733.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::weibull_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4437 of file random.h.

References std::weibull_distribution< _RealType >::operator>(), and std::weibull_distribution< _RealType >::param().

Referenced by std::weibull_distribution< _RealType >::operator().

5.733.3.6 `template<typename _RealType = double> void
std::weibull_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 4415 of file random.h.

5.733.3.7 `template<typename _RealType = double> param_type
std::weibull_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4407 of file random.h.

Referenced by std::weibull_distribution< _RealType >::operator>(), std::operator==(), and std::operator>>().

5.733.3.8 `template<typename _RealType = double> void
std::weibull_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 4386 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.734 `std::weibull_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [weibull_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(1), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.734.1 Detailed Description

`template<typename _RealType = double> struct std::weibull_distribution< _RealType >::param_type`

Parameter type.

Definition at line 4344 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

Chapter 6

File Documentation

6.1 algo.h File Reference

Parallel STL function calls corresponding to the [stl_algo.h](#) header.

Classes

- struct [std::__parallel::__CRandNumber< _MustBeInt >](#)
Functor wrapper for std::rand().

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- template<typename _RAIter >
_RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _FIterator, typename _IteratorTag >
_FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >
_FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)

- `template<typename _RAIter, typename _BinaryPredicate >`
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter`
`__end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type std::__parallel::__count_if -`
`switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random -`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu -`
`parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch`
`(_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type std::__parallel::__count -`
`switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random -`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu -`
`parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_switch (-`
`_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag >`
`_RAIter std::__parallel::__find_first_of_switch (_RAIter __begin1, _RAIter`
`__end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp,`
`random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_Iter std::__parallel::__find_first_of_switch (_Iter __begin1, _Iter __`
`end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _`
`IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`
`>`
`_Iter std::__parallel::__find_first_of_switch (_Iter __begin1, _Iter __end1,`
`_FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter std::__parallel::__find_if_switch (_Iter __begin, _Iter __end, _`
`Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::__parallel::__find_if_switch (_RAIter __begin, _RAIter __end,`
`_Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter std::__parallel::__find_switch (_Iter __begin, _Iter __end, const _Tp`
`&__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter std::__parallel::__find_switch (_RAIter __begin, _RAIter __end,`
`const _Tp &__val, random_access_iterator_tag)`

- `template<typename _Iter , typename _Function , typename _IteratorTag >`
`_Function std::parallel::for_each_switch (_Iter __begin, _Iter __end, _-`
`Function __f, _IteratorTag)`
- `template<typename _RAIter , typename _Function >`
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter _-`
`__end, _Function __f, random_access_iterator_tag, gnu_parallel::Parallelism _-`
`__parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _OutputIterator , typename _Size , typename _Generator , typename _-`
`IteratorTag >`
`_OutputIterator std::parallel::generate_n_switch (_OutputIterator __-`
`begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Size , typename _Generator >`
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n,`
`_Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism _-`
`__parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Generator , typename _IteratorTag >`
`void std::parallel::generate_switch (_FIterator __begin, _FIterator __end,`
`_Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Generator >`
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _-`
`Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism _-`
`__parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`
`_FIterator std::parallel::max_element_switch (_FIterator __begin, _-`
`FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter`
`__end, _Compare __comp, random_access_iterator_tag, gnu_parallel::-`
`Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare ,`
`typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 _-`
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare`
`__comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare >`
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 _-`
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare`
`__comp, random_access_iterator_tag, random_access_iterator_tag, random_-`
`access_iterator_tag)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`
`_FIterator std::parallel::min_element_switch (_FIterator __begin, _-`
`FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter`

```

__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

```

- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`
`_FIterator std::__parallel::__partition_switch (_FIterator __begin, _FIterator`
`__end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::__parallel::__partition_switch (_RAIter __begin, _RAIter __`
`end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`
`void std::__parallel::__replace_if_switch (_FIterator __begin, _FIterator __`
`end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void std::__parallel::__replace_if_switch (_RAIter __begin, _RAIter __`
`end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_`
`tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_`
`balanced)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`
`void std::__parallel::__replace_switch (_FIterator __begin, _FIterator __end,`
`const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`void std::__parallel::__replace_switch (_RAIter __begin, _RAIter __end,`
`const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_`
`tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_`
`balanced)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate,`
`typename _IteratorTag >`
`_FIterator std::__parallel::__search_n_switch (_FIterator __begin, _FIterator`
`__end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`
`_IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter std::__parallel::__search_n_switch (_RAIter __begin, _RAIter __`
`end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`
`random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 std::__parallel::__search_switch (_RAIter1 __begin1, _RAIter1 __`
`end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag,`
`random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename`
`_IteratorTag2 >`
`_FIterator1 std::__parallel::__search_switch (_FIterator1 __begin1, _`
`FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1,`
`_IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 std::__parallel::__search_switch (_RAIter1 __begin1, _RAIter1`

- __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >
_FIterator1 **std::__parallel::search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **std::__parallel::set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **std::__parallel::set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **std::__parallel::set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

- `template<typename _RAIter1, typename _RAIter2, typename _OutputRAIter, typename _Predicate >`
`_OutputRAIter std::__parallel::__set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`
`_RAIter2 std::__parallel::__transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_RAIter2 std::__parallel::__transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`
`_RAIter3 std::__parallel::__transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`
`_OutputIterator std::__parallel::__transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::__parallel::__unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`
`RandomAccessOutputIterator std::__parallel::__unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::__adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator std::__parallel::__adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::__adjacent_find (_FIterator __begin, _FIterator __end)`

- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __-`
`end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __-`
`begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __-`
`begin, _Iter __end, const _Tp &__value, __gnu_parallel::_Parallelism __-`
`parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __-`
`begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __-`
`begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __-`
`begin, _Iter __end, _Predicate __pred, __gnu_parallel::_Parallelism __-`
`parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __-`
`begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, __-`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`
`__begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`
`__begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`
`__begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _-`
`FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu-`
`parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred,`
`__gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iterator, typename _Function >`
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`
`Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function`
`__f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`
`Function __f)`
- `template<typename _FIterator, typename _Generator >`
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`
`Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`
`Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`
`Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`
`_n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`
`_n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`
`_n, _Generator __gen)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __-`
`end)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __-`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`
`Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`
`Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`
`Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end)`

- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, _Compare __comp)`
- `template<typename _FIterator, typename _Predicate >`
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred)`
- `template<typename _FIterator, typename _Predicate >`
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp`
`&__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Tp >`
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _-`
`Tp &__old_value, const _Tp &__new_value, __gnu_parallel::Parallelism _-`
`parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp`
`&__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, const _Tp &__new_value, __gnu_parallel::Parallelism __-`
`parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`
`pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_`
`parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`
`pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_difference (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __`
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __`
`pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _`
`Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _`
`Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`
`__comp, __gnu_parallel::sequential_tag)`

- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`
`__comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`
`__comp)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __-`
`end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_`
`parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`
`BinaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter1 __begin1, _IIter1 __end1,`
`_IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end, _`
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`
`BinaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter1 __begin1, _IIter1 __end1,`
`_IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __`
`gnu_parallel::sequential_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end,`
`_OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::`
`Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`
`BinaryOperation >`

```

_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1,
_Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)
• template<typename _Iter, typename _OutputIterator >
_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
_OutputIterator __out)
• template<typename _Iter, typename _OutputIterator, typename _Predicate >
_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
_OutputIterator __out, _Predicate __pred)
• template<typename _Iter, typename _OutputIterator >
_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
_OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OutputIterator, typename _Predicate >
_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
_OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)

```

6.1.1 Detailed Description

Parallel STL function calls corresponding to the [stl_algo.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algo.h](#).

6.2 `algbase.h` File Reference

Parallel STL function calls corresponding to the [stl_algbase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >
bool **std::__parallel::__lexicographical_compare_switch** (_Iter1 __begin1,

```

    _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _
    IteratorTag1, _IteratorTag2)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
  bool std::__parallel::__lexicographical_compare_switch (_RAIter1 __-
  begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate
  __pred, random_access_iterator_tag, random_access_iterator_tag)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
  pair< _RAIter1, _RAIter2 > std::__parallel::__mismatch_switch (_RAIter1
  __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_
  access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,
  typename _IteratorTag2 >
  pair< _Iter1, _Iter2 > std::__parallel::__mismatch_switch (_Iter1 __-
  begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _
  IteratorTag2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __-
  begin2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  _Predicate __pred)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-
  end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-
  end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-
  end1, _Iter2 __begin2, _Iter2 __end2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_
  parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Predicate __pred)

```


- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`
`__end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`
`__end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`

6.2.1 Detailed Description

Parallel STL function calls corresponding to the [stl_algobase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algobase.h](#).

6.3 algorithm File Reference

6.3.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [algorithm](#).

6.4 algorithm File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > __gnu_cxx::__copy_n (_InputIterator`
`__first, _Size __count, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`
`pair< _RAIterator, _OutputIterator > __gnu_cxx::__copy_n (_RAIterator __`
`first, _Size __count, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

- `int __gnu_cxx::__lexicographical_compare_3way` (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)
- `int __gnu_cxx::__lexicographical_compare_3way` (const char *__first1, const char *__last1, const char *__first2, const char *__last2)
- `template<typename _Tp, typename _Compare >`
`const _Tp & __gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- `template<typename _Tp >`
`const _Tp & __gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`
`_RandomAccessIterator __gnu_cxx::random_sample` (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`
`_RandomAccessIterator __gnu_cxx::random_sample` (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair<_InputIterator, _OutputIterator > __gnu_cxx::copy_n` (_InputIterator __first, _Size __count, _OutputIterator __result)
- `template<typename _InputIterator, typename _Tp, typename _Size >`
`void __gnu_cxx::count` (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- `template<typename _InputIterator, typename _Predicate, typename _Size >`
`void __gnu_cxx::count_if` (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- `template<typename _RandomAccessIterator >`
`bool __gnu_cxx::is_heap` (_RandomAccessIterator __first, _RandomAccessIterator __last)
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_heap` (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_sorted` (_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)
- `template<typename _ForwardIterator >`
`bool __gnu_cxx::is_sorted` (_ForwardIterator __first, _ForwardIterator __last)
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way` (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`

```

_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __-
_first, _InputIterator __last, _RandomAccessIterator __out_first, __-
_RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
• template<typename _InputIterator, typename _RandomAccessIterator >
_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __-
_first, _InputIterator __last, _RandomAccessIterator __out_first, __-
_RandomAccessIterator __out_last)
• template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename
_RandomNumberGenerator >
_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, __-
_FowardIterator __last, _OutputIterator __out, const _Distance __n, __-
_RandomNumberGenerator &__rand)
• template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >
_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, __-
_FowardIterator __last, _OutputIterator __out, const _Distance __n)

```

6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/algorithm](#).

6.5 algorithm File Reference

6.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [parallel/algorithm](#).

6.6 algorithmfwd.h File Reference

Namespaces

- namespace [std](#)

Functions

- template<typename _Filter >
_Filter **std::adjacent_find** (_Filter, _Filter)

- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`
`bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`
`bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2 >`
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _-`
`Predicate)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`
`BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _-`
`Compare)`
- `template<typename _Filter, typename _Tp >`
`void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`
`_OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`

- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`

- `template<typename _Iter, typename _Predicate >`
`_Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`
`_Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`
`void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter >`
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`
`void std::iter_swap (_Filter1, _Filter2)`

- `template<typename _Iter1, typename _Iter2 >`
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b)`

- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _FIter >`
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter, _Compare)`
- `template<typename _IIter1, typename _IIter2 >`
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate >`
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2, _BinaryPredicate)`
- `template<typename _BIter >`
`bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _IIter, typename _Predicate >`
`bool std::none_of (_IIter, _IIter, _Predicate)`
- `template<typename _RAIter >`
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter, typename _Compare >`
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter >`
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter)`
- `template<typename _BIter, typename _Predicate >`
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > std::partition_copy (_IIter, _IIter, _OIter1, _OIter2, _Predicate)`
- `template<typename _FIter, typename _Predicate >`
`_FIter std::partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _RAIter >`
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::pop_heap (_RAIter, _RAIter, _Compare)`

- `template<typename _BIter >`
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter >`
`void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter, typename _Generator >`
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RAIter >`
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _FIter, typename _Tp >`
`_FIter std::remove (_FIter, _FIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`
`_OIter std::remove_copy (_IIter, _IIter, _OIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`
`_OIter std::remove_copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`
`_FIter std::remove_if (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Tp >`
`void std::replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`
`_OIter std::replace_copy (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter std::replace_copy_if (_IIter, _IIter, _OIter, _Predicate, const _Tp &)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void std::replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `template<typename _BIter >`
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _FIter >`
`void std::rotate (_FIter, _FIter, _FIter)`
- `template<typename _FIter, typename _OIter >`
`_OIter std::rotate_copy (_FIter, _FIter, _FIter, _OIter)`
- `template<typename _FIter1, typename _FIter2 >`
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate >`
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp >`
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _RAIter, typename _UGenerator >`
`void std::shuffle (_RAIter, _RAIter, _UGenerator &)`
- `template<typename _RAIter, typename _Compare >`
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter >`
`void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter, typename _Compare >`
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _Tp, size_t _Nm >`
`void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`
`_OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`

- `template<typename _Filter >`
`_Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`
`_OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

6.6.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/algorithmfwd.h](#).

6.7 algorithmfwd.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- `template<typename _Filter, typename _IterTag >`
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _RAIter >`
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _RAIter, typename _BiPredicate >`
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`

- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _IIter, typename _Tp, typename _IterTag >`
`iterator_traits< _IIter >::difference_type std::parallel::count_switch (_IIter, _IIter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _IIter, typename _FIter, typename _IterTag1, typename _IterTag2 >`
`_IIter std::parallel::find_first_of_switch (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag >`
`_RAIter std::parallel::find_first_of_switch (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _IIter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_IIter std::parallel::find_first_of_switch (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _IIter, typename _Predicate, typename _IterTag >`
`_IIter std::parallel::find_if_switch (_IIter, _IIter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _IIter, typename _Tp, typename _IterTag >`
`_IIter std::parallel::find_switch (_IIter, _IIter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _IIter, typename _Function, typename _IterTag >`
`_Function std::parallel::for_each_switch (_IIter, _IIter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function >`
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n,`

- ```

 _Generator __gen, random_access_iterator_tag, __gnu_parallel::Parallelism _
 _parallelism_tag=__gnu_parallel::parallel_balanced)

```
- `template<typename _Filter, typename _Generator, typename _IterTag >`  
`void std::__parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
  - `template<typename _RAIter, typename _Generator >`  
`void std::__parallel::generate_switch (_RAIter __begin, _RAIter __end, _`  
`Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _`  
`parallelism_tag=__gnu_parallel::parallel_balanced)`
  - `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`bool std::__parallel::lexicographical_compare_switch (_Iter1, _Iter1, _`  
`Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare_switch (_RAIter1 __`  
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate`  
`__pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::__parallel::max_element_switch (_Filter, _Filter, _Compare, _`  
`IterTag)`
  - `template<typename _RAIter, typename _Compare >`  
`_RAIter std::__parallel::max_element_switch (_RAIter __begin, _RAIter`  
`__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
  - `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename`  
`_IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::__parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _`  
`OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2,`  
`_OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
  - `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::__parallel::min_element_switch (_Filter, _Filter, _Compare, _`  
`IterTag)`
  - `template<typename _RAIter, typename _Compare >`  
`_RAIter std::__parallel::min_element_switch (_RAIter __begin, _RAIter`  
`__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::__parallel::mismatch_switch (_RAIter1`  
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random-`  
`access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`

- 
- ```
pair<_Iter1, _Iter2> std::parallel::mismatch_switch (_Iter1, _Iter1,
    _Iter2, _Predicate, _IterTag1, _IterTag2)
```
- `template<typename _Filter, typename _Predicate, typename _IterTag >`
`_Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _-`
`IterTag)`
 - `template<typename _RAIter, typename _Predicate >`
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __-`
`end, _Predicate __pred, random_access_iterator_tag)`
 - `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _-`
`Tp &, _IterTag)`
 - `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __-`
`end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_`
`tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_`
`balanced)`
 - `template<typename _Filter, typename _Tp, typename _IterTag >`
`void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp`
`&, _IterTag)`
 - `template<typename _RAIter, typename _Tp >`
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end,`
`const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_`
`tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_`
`balanced)`
 - `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer,`
`const _Tp &, _BiPredicate, random_access_iterator_tag)`
 - `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename`
`_IterTag >`
`_Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp`
`&, _BiPredicate, _IterTag)`
 - `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 _-`
`__end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag,`
`random_access_iterator_tag)`
 - `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2,`
`_IterTag1, _IterTag2)`
 - `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`
`_RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _-`
`RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_`
`tag)`
 - `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1,`
`typename _IterTag2 >`
-

```

_Filter1 std::__parallel::__search_switch (_Filter1, _Filter1, _Filter2, _Filter2,
_BiPredicate, _IterTag1, _IterTag2)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_difference_switch (_Iter1, _Iter1, _Iter2, _-
Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename
_Predicate >
_Output_RAIter std::__parallel::__set_difference_switch (_RAIter1 _-
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _-
Output_RAIter __result, _Predicate __pred, random_access_iterator_tag,
random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_intersection_switch (_Iter1, _Iter1, _Iter2, _-
Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _-
Predicate >
_Output_RAIter std::__parallel::__set_intersection_switch (_RAIter1 _-
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_-
RAIter __result, _Predicate __pred, random_access_iterator_tag, random_-
access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_symmetric_difference_switch (_Iter1, _Iter1,
_Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename
_Predicate >
_Output_RAIter std::__parallel::__set_symmetric_difference_switch (_-
RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2,
_Output_RAIter __result, _Predicate __pred, random_access_iterator_tag,
random_access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2,
_OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _-
Predicate >
_Output_RAIter std::__parallel::__set_union_switch (_RAIter1 __begin1, _-
RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter
__result, _Predicate __pred, random_access_iterator_tag, random_access_-
iterator_tag, random_access_iterator_tag)
• template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1,
typename _IterTag2 >
_OIter std::__parallel::__transform1_switch (_Iter, _Iter, _OIter, _-
UnaryOperation, _IterTag1, _IterTag2)

```

- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`
`_RAOIter std::__parallel::__transform1_switch (_RAIter, _RAIter,`
`_RAOIter, _UnaryOperation, random_access_iterator_tag, random_`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism=__gnu_`
`parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation`
`>`
`_RAIter3 std::__parallel::__transform2_switch (_RAIter1, _RAIter1, _`
`RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_`
`access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism`
`__parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, type-`
`name _Tag1, typename _Tag2, typename _Tag3 >`
`_OIter std::__parallel::__transform2_switch (_Iter1, _Iter1, _Iter2, _OIter,`
`_BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, type-`
`name _IterTag2 >`
`_OIter std::__parallel::__unique_copy_switch (_Iter, _Iter, _OIter, _`
`Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`
`_RandomAccess_OIter std::__parallel::__unique_copy_switch (_RAIter,`
`_RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag,`
`random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter std::__parallel::__adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter >`
`_Filter std::__parallel::__adjacent_find (_Filter, _Filter, __gnu_`
`parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter std::__parallel::__adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_`
`parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter std::__parallel::__adjacent_find (_Filter, _Filter)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count (_Iter __`
`begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count (_Iter __`
`begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count (_Iter __`
`begin, _Iter __end, const _Tp &__value, __gnu_parallel::__Parallelism __`
`parallelism_tag)`

- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __`
`__begin, _Iter __end, _Predicate __pred, __gnu_parallel::_Parallelism __`
`parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __`
`begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __`
`begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`
`begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, -`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _FIter >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, __gnu_`
`parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function`
`__f, __gnu_parallel::sequential_tag)`

- `template<typename _Iterator, typename _Function >`
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`
`Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`
`_Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`
`void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`
`void std::__parallel::generate (_Filter, _Filter, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Generator >`
`void std::__parallel::generate (_Filter, _Filter, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`
`end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`
`end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`
`end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`
`end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`
`_Filter std::__parallel::max_element (_Filter, _Filter, __gnu_parallel::Parallelism)`
- `template<typename _Filter >`
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter std::__parallel::max_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`

- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter std::__parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter std::__parallel::min_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _Filter >`
`_Filter std::__parallel::min_element (_Filter, _Filter, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`

- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate >`
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, __gnu_`
`parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, __`
`gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &&__rand)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Tp >`
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, __`
`gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Tp >`
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`

- `template<typename _Filter, typename _Tp >`
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, __BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, __BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, __BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, __BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`
`Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`
`Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`
`Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`
`Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __-`
`gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`

- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, __gnu_parallel::sequential_tag)`

6.7.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/algorithmfwd.h](#).

6.8 allocator.h File Reference

Classes

- class [std::allocator<_Tp>](#)
The standard allocator, as per [20.4].
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.
- class [std::allocator<void>](#)
[allocator<void>](#) specialization.

Namespaces

- namespace [std](#)

Functions

- `template<typename _T1, typename _T2>
bool std::operator!=(const allocator<_T1> &, const allocator<_T2> &)`
- `template<typename _Tp>
bool std::operator!=(const allocator<_Tp> &, const allocator<_Tp> &)`
- `template<typename _T1, typename _T2>
bool std::operator==(const allocator<_T1> &, const allocator<_T2> &)`
- `template<typename _Tp>
bool std::operator==(const allocator<_Tp> &, const allocator<_Tp> &)`

6.8.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [allocator.h](#).

6.9 array File Reference

6.9.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [array](#).

6.10 array File Reference

Classes

- struct [std::tr1::array<_Tp, _Nm>](#)
A standard container for storing a fixed size sequence of elements.

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>
_Tp & std::tr1::get (array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>
const _Tp & std::tr1::get (const array< _Tp, _Nm > &__arr)`
- `template<typename _Tp, std::size_t _Nm>
bool std::tr1::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>
bool std::tr1::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>
bool std::tr1::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>
bool std::tr1::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>
bool std::tr1::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>
bool std::tr1::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>
void std::tr1::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`

6.10.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/array](#).

6.11 array_allocator.h File Reference

Classes

- class [__gnu_cxx::array_allocator< _Tp, _Array >](#)
An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class [__gnu_cxx::array_allocator_base< _Tp >](#)
Base class.

Namespaces

- namespace [__gnu_cxx](#)

Functions

- [template<typename _Tp, typename _Array >](#)
[bool __gnu_cxx::operator!=](#) (const [array_allocator< _Tp, _Array >](#) &, const [array_allocator< _Tp, _Array >](#) &)
- [template<typename _Tp, typename _Array >](#)
[bool __gnu_cxx::operator==](#) (const [array_allocator< _Tp, _Array >](#) &, const [array_allocator< _Tp, _Array >](#) &)

6.11.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [array_allocator.h](#).

6.12 `assoc_container.hpp` File Reference

Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >`

An abstract basic hash-based associative container.

- class `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`

An abstract basic tree-like (tree, trie) associative container.

- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`

A concrete collision-chaining hash-based associative container.

- class `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`

An abstract basic associative container.

- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`

A concrete general-probing hash-based associative container.

- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`

A list-update based associative container.

- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`

A concrete basic tree-based associative container.

- class `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`

A concrete basic trie-based associative container.

Namespaces

- namespace `__gnu_pbds`

Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`

6.12.1 Detailed Description

Contains associative containers.

Definition in file [assoc_container.hpp](#).

6.13 atomic File Reference

Classes

- struct [std::atomic< _Tp >](#)
atomic /// 29.4.3, Generic atomic type, primary class template.
- struct [std::atomic< _Tp * >](#)
Partial specialization for pointer types.
- struct [std::atomic< bool >](#)
Explicit specialization for bool.
- struct [std::atomic< char >](#)
Explicit specialization for char.
- struct [std::atomic< char16_t >](#)
Explicit specialization for char16_t.
- struct [std::atomic< char32_t >](#)

Explicit specialization for char32_t.

- struct `std::atomic< int >`
Explicit specialization for int.
- struct `std::atomic< long >`
Explicit specialization for long.
- struct `std::atomic< long long >`
Explicit specialization for long long.
- struct `std::atomic< short >`
Explicit specialization for short.
- struct `std::atomic< signed char >`
Explicit specialization for signed char.
- struct `std::atomic< unsigned char >`
Explicit specialization for unsigned char.
- struct `std::atomic< unsigned int >`
Explicit specialization for unsigned int.
- struct `std::atomic< unsigned long >`
Explicit specialization for unsigned long.
- struct `std::atomic< unsigned long long >`
Explicit specialization for unsigned long long.
- struct `std::atomic< unsigned short >`
Explicit specialization for unsigned short.
- struct `std::atomic< void * >`
Explicit specialization for void.*
- struct `std::atomic< wchar_t >`
Explicit specialization for wchar_t.

Namespaces

- namespace `std`

Functions

- memory_order **std::__calculate_memory_order** (memory_order __m)
- template<typename _ITp >
bool **std::atomic_compare_exchange_strong** (__atomic_base< _ITp > *__a,
_ITp *__i1, _ITp __i2)
- bool **std::atomic_compare_exchange_strong** (atomic_address *__a, void **__v1, void *__v2)
- bool **std::atomic_compare_exchange_strong** (atomic_bool *__a, bool *__i1, bool __i2)
- template<typename _ITp >
bool **std::atomic_compare_exchange_strong_explicit** (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)
- bool **std::atomic_compare_exchange_strong_explicit** (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **std::atomic_compare_exchange_strong_explicit** (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)
- template<typename _ITp >
bool **std::atomic_compare_exchange_weak** (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)
- bool **std::atomic_compare_exchange_weak** (atomic_address *__a, void **__v1, void *__v2)
- bool **std::atomic_compare_exchange_weak** (atomic_bool *__a, bool *__i1, bool __i2)
- template<typename _ITp >
bool **std::atomic_compare_exchange_weak_explicit** (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)
- bool **std::atomic_compare_exchange_weak_explicit** (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **std::atomic_compare_exchange_weak_explicit** (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)
- void * **std::atomic_exchange** (atomic_address *__a, void *__v)
- template<typename _ITp >
_ITp **std::atomic_exchange** (__atomic_base< _ITp > *__a, _ITp __i)
- bool **std::atomic_exchange** (atomic_bool *__a, bool __i)
- template<typename _ITp >
_ITp **std::atomic_exchange_explicit** (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)
- void * **std::atomic_exchange_explicit** (atomic_address *__a, void *__v, memory_order __m)
- bool **std::atomic_exchange_explicit** (atomic_bool *__a, bool __i, memory_order __m)
- void * **std::atomic_fetch_add** (atomic_address *__a, ptrdiff_t __d)

- `template<typename _ITp >`
`_ITp std::atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m)`
- `bool std::atomic_is_lock_free (const atomic_bool *__a)`
- `bool std::atomic_is_lock_free (const atomic_address *__a)`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `void * std::atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`
`_ITp std::atomic_load (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_load (const atomic_bool *__a)`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const __atomic_base< _ITp > *__a,`
`memory_order __m)`

- void * **std::atomic_load_explicit** (const atomic_address *__a, memory_order __m)
- bool **std::atomic_load_explicit** (const atomic_bool *__a, memory_order __m)
- void **std::atomic_store** (atomic_bool *__a, bool __i)
- template<typename _ITp >
void **std::atomic_store** (__atomic_base< _ITp > *__a, _ITp __i)
- void **std::atomic_store** (atomic_address *__a, void *__v)
- void **std::atomic_store_explicit** (atomic_bool *__a, bool __i, memory_order __m)
- void **std::atomic_store_explicit** (atomic_address *__a, void *__v, memory_order __m)
- template<typename _ITp >
void **std::atomic_store_explicit** (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)
- template<typename _Tp >
_Tp **std::kill_dependency** (_Tp __y)

6.13.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic](#).

6.14 atomic_0.h File Reference

Classes

- struct [__atomic0::atomic_address](#)
29.4.2, address types
- struct [__atomic0::atomic_bool](#)
atomic_bool
- struct [__atomic0::atomic_flag](#)
atomic_flag

Defines

- #define **_ATOMIC_CMPEXCHNG_**(__a, __e, __m, __x)
- #define **_ATOMIC_LOAD_**(__a, __x)
- #define **_ATOMIC_MODIFY_**(__a, __o, __m, __x)
- #define **_ATOMIC_STORE_**(__a, __m, __x)

6.14.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic_0.h](#).

6.15 atomic_2.h File Reference

Classes

- struct [__atomic2::atomic_address](#)
29.4.2, address types
- struct [__atomic2::atomic_bool](#)
atomic_bool
- struct [__atomic2::atomic_flag](#)
atomic_flag

6.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic_2.h](#).

6.16 atomic_base.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _ATOMIC_CMPEXCHNG_(__a, __e, __m, __x)`
- `#define _ATOMIC_LOAD_(__a, __x)`
- `#define _ATOMIC_MODIFY_(__a, __o, __m, __x)`
- `#define _ATOMIC_STORE_(__a, __m, __x)`
- `#define _GLIBCXX_ATOMIC_BASE_H`

- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

Typedefs

- typedef struct std::__atomic_flag_base **std::__atomic_flag_base**
- typedef `atomic_short` **std::atomic_int_fast16_t**
- typedef `atomic_int` **std::atomic_int_fast32_t**
- typedef `atomic_llong` **std::atomic_int_fast64_t**
- typedef `atomic_schar` **std::atomic_int_fast8_t**
- typedef `atomic_short` **std::atomic_int_least16_t**
- typedef `atomic_int` **std::atomic_int_least32_t**
- typedef `atomic_llong` **std::atomic_int_least64_t**
- typedef `atomic_schar` **std::atomic_int_least8_t**
- typedef `atomic_llong` **std::atomic_intmax_t**
- typedef `atomic_long` **std::atomic_intptr_t**
- typedef `atomic_long` **std::atomic_ptrdiff_t**
- typedef `atomic_ulong` **std::atomic_size_t**
- typedef `atomic_long` **std::atomic_ssize_t**
- typedef `atomic_ushort` **std::atomic_uint_fast16_t**
- typedef `atomic_uint` **std::atomic_uint_fast32_t**
- typedef `atomic_ullong` **std::atomic_uint_fast64_t**
- typedef `atomic_uchar` **std::atomic_uint_fast8_t**
- typedef `atomic_ushort` **std::atomic_uint_least16_t**
- typedef `atomic_uint` **std::atomic_uint_least32_t**
- typedef `atomic_ullong` **std::atomic_uint_least64_t**
- typedef `atomic_uchar` **std::atomic_uint_least8_t**
- typedef `atomic_ullong` **std::atomic_uintmax_t**
- typedef `atomic_ulong` **std::atomic_uintptr_t**
- typedef enum `std::memory_order` **std::memory_order**

Enumerations

- enum `std::memory_order` {
 memory_order_relaxed, **memory_order_consume**, **memory_order_-**
 acquire, **memory_order_release**,
 memory_order_acq_rel, **memory_order_seq_cst** }

Functions

- void **std::__atomic_flag_wait_explicit** (__atomic_flag_base *, memory_order) _GLIBCXX_NOTHROW
- **std::__attribute__** ((__const__)) __atomic_flag_base *__atomic_flag_for_address(const void *__z) _GLIBCXX_NOTHROW
- void **std::atomic_flag_clear** (__atomic_flag_base *__a)
- void **std::atomic_flag_clear_explicit** (__atomic_flag_base *, memory_order) _GLIBCXX_NOTHROW
- bool **std::atomic_flag_test_and_set** (__atomic_flag_base *__a)
- bool **std::atomic_flag_test_and_set_explicit** (__atomic_flag_base *, memory_order) _GLIBCXX_NOTHROW

6.16.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic_base.h](#).

6.17 atomic_word.h File Reference

Typedefs

- typedef int **_Atomic_word**

6.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomic_word.h](#).

6.18 atomicfwd_c.h File Reference

Defines

- #define **_ATOMIC_MEMBER_**
- #define **atomic_compare_exchange**(__a, __e, __m)
- #define **atomic_compare_exchange_explicit**(__a, __e, __m, __x, __y)
- #define **atomic_exchange**(__a, __m)
- #define **atomic_exchange_explicit**(__a, __m, __x)
- #define **atomic_fetch_add**(__a, __m)

- `#define atomic_fetch_add_explicit(__a, __m, __x)`
- `#define atomic_fetch_and(__a, __m)`
- `#define atomic_fetch_and_explicit(__a, __m, __x)`
- `#define atomic_fetch_or(__a, __m)`
- `#define atomic_fetch_or_explicit(__a, __m, __x)`
- `#define atomic_fetch_sub(__a, __m)`
- `#define atomic_fetch_sub_explicit(__a, __m, __x)`
- `#define atomic_fetch_xor(__a, __m)`
- `#define atomic_fetch_xor_explicit(__a, __m, __x)`
- `#define atomic_is_lock_free(__a)`
- `#define atomic_load(__a)`
- `#define atomic_load_explicit(__a, __x)`
- `#define atomic_store(__a, __m)`
- `#define atomic_store_explicit(__a, __m, __x)`

Typedefs

- `typedef struct __atomic_address_base atomic_address`
- `typedef struct __atomic_bool_base atomic_bool`
- `typedef struct __atomic_char_base atomic_char`
- `typedef struct __atomic_short_base atomic_char16_t`
- `typedef struct __atomic_int_base atomic_char32_t`
- `typedef struct __atomic_flag_base atomic_flag`
- `typedef struct __atomic_int_base atomic_int`
- `typedef struct __atomic_llong_base atomic_llong`
- `typedef struct __atomic_long_base atomic_long`
- `typedef struct __atomic_schar_base atomic_schar`
- `typedef struct __atomic_short_base atomic_short`
- `typedef struct __atomic_uchar_base atomic_uchar`
- `typedef struct __atomic_uint_base atomic_uint`
- `typedef struct __atomic_ullong_base atomic_ullong`
- `typedef struct __atomic_ulong_base atomic_ulong`
- `typedef struct __atomic_ushort_base atomic_ushort`
- `typedef struct __atomic_wchar_t_base atomic_wchar_t`

6.18.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd_c.h](#).

6.19 atomicfwd_cxx.h File Reference

Defines

- #define `_ATOMIC_MEMBER_`

Typedefs

- typedef `__atomic_base< char >` [atomic_char](#)
- typedef `__atomic_base< char16_t >` [atomic_char16_t](#)
- typedef `__atomic_base< char32_t >` [atomic_char32_t](#)
- typedef `__atomic_base< int >` [atomic_int](#)
- typedef `__atomic_base< long long >` [atomic_llong](#)
- typedef `__atomic_base< long >` [atomic_long](#)
- typedef `__atomic_base< signed char >` [atomic_schar](#)
- typedef `__atomic_base< short >` [atomic_short](#)
- typedef `__atomic_base< unsigned char >` [atomic_uchar](#)
- typedef `__atomic_base< unsigned int >` [atomic_uint](#)
- typedef `__atomic_base< unsigned long long >` [atomic_ullong](#)
- typedef `__atomic_base< unsigned long >` [atomic_ulong](#)
- typedef `__atomic_base< unsigned short >` [atomic_ushort](#)
- typedef `__atomic_base< wchar_t >` [atomic_wchar_t](#)

6.19.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd_cxx.h](#).

6.20 atomicity.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Defines

- #define `_GLIBCXX_READ_MEM_BARRIER`
- #define `_GLIBCXX_WRITE_MEM_BARRIER`

Functions

- static void `__gnu_cxx::__atomic_add` (volatile `_Atomic_word` *`__mem`, int `__val`)
- static void `__gnu_cxx::__atomic_add_single` (`_Atomic_word` *`__mem`, int `__val`)
- static `_Atomic_word` `__gnu_cxx::__attribute__` ((__unused__) `__exchange_and_add_dispatch`(`_Atomic_word` *`__mem`
- static `_Atomic_word` `__gnu_cxx::__exchange_and_add` (volatile `_Atomic_word` *`__mem`, int `__val`)
- else return `__gnu_cxx::__exchange_and_add_single` (`__mem`, `__val`)
- static `_Atomic_word` `__gnu_cxx::__exchange_and_add_single` (`_Atomic_word` *`__mem`, int `__val`)
- static `_Atomic_word` int `__val` `__gnu_cxx::if` (`__gthread_active_p`()) return `__exchange_and_add`(`__mem`

Variables

- static `_Atomic_word` int `__val` `__gnu_cxx::__val`

6.20.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicity.h](#).

6.21 auto_ptr.h File Reference

Classes

- class [std::auto_ptr<_Tp>](#)
A simple smart pointer providing strict ownership semantics.
- struct [std::auto_ptr_ref<_Tp1>](#)

Namespaces

- namespace [std](#)

6.21.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [auto_ptr.h](#).

6.22 `balanced_quicksort.h` File Reference

Implementation of a dynamically load-balanced parallel quicksort.

Classes

- struct [__gnu_parallel::__QSBThreadLocal](#)< [_RAIter](#) >
Information local to one thread in the parallel quicksort run.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- template<typename [_RAIter](#) , typename [_Compare](#) >
void [__gnu_parallel::__parallel_sort_qsb](#) ([_RAIter](#) __begin, [_RAIter](#) __end, [_Compare](#) __comp, [_ThreadIndex](#) __num_threads)
- template<typename [_RAIter](#) , typename [_Compare](#) >
void [__gnu_parallel::__qsb_conquer](#) ([_QSBThreadLocal](#)< [_RAIter](#) > *__tls, [_RAIter](#) __begin, [_RAIter](#) __end, [_Compare](#) __comp, [_ThreadIndex](#) __iam, [_ThreadIndex](#) __num_threads, bool __parent_wait)
- template<typename [_RAIter](#) , typename [_Compare](#) >
[std::iterator_traits](#)< [_RAIter](#) >::difference_type [__gnu_parallel::__qsb_divide](#) ([_RAIter](#) __begin, [_RAIter](#) __end, [_Compare](#) __comp, [_ThreadIndex](#) __num_threads)
- template<typename [_RAIter](#) , typename [_Compare](#) >
void [__gnu_parallel::__qsb_local_sort_with_helping](#) ([_QSBThreadLocal](#)< [_RAIter](#) > *__tls, [_Compare](#) &__comp, [_ThreadIndex](#) __iam, bool __wait)

6.22.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort. It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [balanced_quicksort.h](#).

6.23 base.h File Reference

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- class [__gnu_parallel::__binder1st](#)< [_Operation](#), [_FirstArgumentType](#), [_SecondArgumentType](#), [_ResultType](#) >
Similar to [std::binder1st](#), but giving the argument types explicitly.
- class [__gnu_parallel::__binder2nd](#)< [_Operation](#), [_FirstArgumentType](#), [_SecondArgumentType](#), [_ResultType](#) >
Similar to [std::binder2nd](#), but giving the argument types explicitly.
- class [__gnu_parallel::__unary_negate](#)< [_Predicate](#), [argument_type](#) >
Similar to [std::unary_negate](#), but giving the argument types explicitly.
- class [__gnu_parallel::__EqualFromLess](#)< [_T1](#), [_T2](#), [_Compare](#) >
Constructs predicate for equality from strict weak ordering predicate.
- struct [__gnu_parallel::__EqualTo](#)< [_T1](#), [_T2](#) >
Similar to [std::equal_to](#), but allows two different types.
- struct [__gnu_parallel::__Less](#)< [_T1](#), [_T2](#) >
Similar to [std::less](#), but allows two different types.
- struct [__gnu_parallel::__Multiplies](#)< [_Tp1](#), [_Tp2](#), [_Result](#) >
Similar to [std::multiplies](#), but allows two different types.

- struct `__gnu_parallel::Plus<_Tp1, _Tp2, _Result >`
Similar to `std::plus`, but allows two different types.
- class `__gnu_parallel::PseudoSequence<_Tp, _DifferenceTp >`
Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.
- class `__gnu_parallel::PseudoSequenceIterator<_Tp, _DifferenceTp >`
Iterator associated with `__gnu_parallel::PseudoSequence`. It features the usual random-access iterator functionality.

Namespaces

- namespace `__gnu_parallel`
- namespace `__gnu_sequential`
- namespace `std`
- namespace `std::__parallel`

Defines

- `#define GLIBCXX_PARALLEL_ASSERT(_Condition)`

Functions

- void `__gnu_parallel::__decode2` (`_CASable __x`, `int &__a`, `int &__b`)
- `_CASable` `__gnu_parallel::__encode2` (`int __a`, `int __b`)
- `_ThreadIndex` `__gnu_parallel::__get_max_threads` ()
- bool `__gnu_parallel::__is_parallel` (`const _Parallelism __p`)
- template<typename `_RAIter`, typename `_Compare` >
`_RAIter` `__gnu_parallel::__median_of_three_iterators` (`_RAIter __a`, `_RAIter __b`, `_RAIter __c`, `_Compare __comp`)
- template<typename `_Size` >
`_Size` `__gnu_parallel::__rd_log2` (`_Size __n`)
- template<typename `_Tp` >
`const _Tp &` `__gnu_parallel::max` (`const _Tp &__a`, `const _Tp &__b`)
- template<typename `_Tp` >
`const _Tp &` `__gnu_parallel::min` (`const _Tp &__a`, `const _Tp &__b`)

6.23.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/base.h](#).

6.24 `base.h` File Reference

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_profile](#)
- namespace [std](#)
- namespace [std::__profile](#)

6.24.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/base.h](#).

6.25 `basic_file.h` File Reference

Namespaces

- namespace [std](#)

6.25.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic_file.h](#).

6.26 `basic_ios.h` File Reference

Classes

- class `std::basic_ios<_CharT, _Traits>`

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Namespaces

- namespace `std`

Functions

- template<typename _Facet>
const _Facet & **std::__check_facet** (const _Facet *__f)

6.26.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic_ios.h](#).

6.27 `basic_ios.tcc` File Reference

Namespaces

- namespace `std`

6.27.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic_ios.tcc](#).

6.28 `basic_iterator.h` File Reference

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

6.28.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [basic_iterator.h](#).

6.29 `basic_string.h` File Reference

Classes

- class [std::basic_string<_CharT, _Traits, _Alloc>](#)
Managing sequences of characters and character-like objects.
- struct [std::hash<string>](#)
std::hash specialization for string.
- struct [std::hash<u16string>](#)
std::hash specialization for u16string.
- struct [std::hash<u32string>](#)
std::hash specialization for u32string.
- struct [std::hash<wstring>](#)
std::hash specialization for wstring.

Namespaces

- namespace [std](#)

Functions

- template<typename _CharT, typename _Traits, typename _Alloc>
[basic_istream<_CharT, _Traits>](#) & [std::getline](#) ([basic_istream<_CharT, _Traits>](#) & __is, [basic_string<_CharT, _Traits, _Alloc>](#) & __str, _CharT __delim)

- `template<>`
`basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in,`
`basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _-`
`_Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`
`basic_istream< char > & std::getline (basic_istream< char > &__in, basic_-`
`string< char > &__str, char __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _-`
`CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _-`
`CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc`
`> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _-`
`CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _-`
`Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type`
`std::operator== (const basic_string< _CharT > &__lhs, const basic_string<`
`_CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _-`
`Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`
`_Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`
`basic_istream< char > & std::operator>> (basic_istream< char > &__is,`
`basic_string< char > &__str)`
- `double std::stod (const string &__str, size_t *__idx=0)`
- `float std::stof (const string &__str, size_t *__idx=0)`
- `int std::stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `long std::stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long double std::stold (const string &__str, size_t *__idx=0)`
- `long long std::stoll (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long std::stoul (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long long std::stoull (const string &__str, size_t *__idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string<`
`_CharT, _Traits, _Alloc > &__rhs)`
- `string std::to_string (long double __val)`
- `string std::to_string (int __val)`
- `string std::to_string (long __val)`
- `string std::to_string (unsigned __val)`
- `string std::to_string (float __val)`
- `string std::to_string (unsigned long long __val)`
- `string std::to_string (unsigned long __val)`
- `string std::to_string (double __val)`
- `string std::to_string (long long __val)`
- `wstring std::to_wstring (unsigned long __val)`
- `wstring std::to_wstring (long __val)`
- `wstring std::to_wstring (int __val)`
- `wstring std::to_wstring (unsigned __val)`
- `wstring std::to_wstring (unsigned long long __val)`
- `wstring std::to_wstring (long long __val)`
- `wstring std::to_wstring (double __val)`
- `wstring std::to_wstring (long double __val)`
- `wstring std::to_wstring (float __val)`

6.29.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic_string.h](#).

6.30 `basic_string.tcc` File Reference

Namespaces

- namespace `std`

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`

6.30.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic_string.tcc](#).

6.31 `basic_types.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >](#)
- struct [__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >](#)
- struct [__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >](#)
- struct [__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >](#)

Namespaces

- namespace [__gnu_pbds](#)

Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

6.31.1 Detailed Description

Contains basic types used by containers.

Definition in file [basic_types.hpp](#).

6.32 binders.h File Reference

Classes

- class [std::binder1st< _Operation >](#)
One of the [binder functors](#).
- class [std::binder2nd< _Operation >](#)
One of the [binder functors](#).

Namespaces

- namespace [std](#)

Functions

- `template<typename _Operation , typename _Tp >`
`binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &_`
`_x)`
- `template<typename _Operation , typename _Tp >`
`binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &_`
`_x)`

Variables

- [std::binder1st std::_GLIBCXX_DEPRECATED_ATTR](#)

6.32.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [binders.h](#).

6.33 bitmap_allocator.h File Reference

Classes

- class [__gnu_cxx::__detail::__mini_vector< _Tp >](#)
__mini_vector<> is a stripped down version of the full-fledged std::vector<>.
- class [__gnu_cxx::__detail::Bitmap_counter< _Tp >](#)
The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.
- class [__gnu_cxx::__detail::Ffit_finder< _Tp >](#)
The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.
- class [__gnu_cxx::bitmap_allocator< _Tp >](#)
Bitmap Allocator, primary template.
- class [__gnu_cxx::free_list](#)
The free list class for managing chunks of memory to be given to and returned by the [bitmap_allocator](#).

Namespaces

- namespace [__gnu_cxx](#)
- namespace [__gnu_cxx::__detail](#)

Defines

- [#define _BALLOC_ALIGN_BYTES](#)

Enumerations

- enum { **bits_per_byte**, **bits_per_block** }

Functions

- void [__gnu_cxx::__detail::__bit_allocate](#) (size_t *__pmap, size_t __pos) throw ()
- void [__gnu_cxx::__detail::__bit_free](#) (size_t *__pmap, size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
_ForwardIterator [__gnu_cxx::__detail::__lower_bound](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair >
size_t [__gnu_cxx::__detail::__num_bitmaps](#) (_AddrPair __ap)
- template<typename _AddrPair >
size_t [__gnu_cxx::__detail::__num_blocks](#) (_AddrPair __ap)
- size_t [__gnu_cxx::__Bit_scan_forward](#) (size_t __num)
- template<typename _Tp1, typename _Tp2 >
bool [__gnu_cxx::operator!=](#) (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _Tp1, typename _Tp2 >
bool [__gnu_cxx::operator==](#) (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()

6.33.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [bitmap_allocator.h](#).

6.33.2 Define Documentation

6.33.2.1 #define _BALLOC_ALIGN_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 44 of file [bitmap_allocator.h](#).

6.34 bitset File Reference

Classes

- struct [std::_Base_bitset< _Nw >](#)

- struct [std::_Base_bitset< 0 >](#)
- struct [std::_Base_bitset< 1 >](#)
- class [std::bitset< _Nb >](#)

The bitset class represents a fixed-size sequence of bits.

- class [std::bitset< _Nb >::reference](#)
- struct [std::hash<::bitset< _Nb > >](#)

[std::hash](#) specialization for bitset.

Namespaces

- namespace [std](#)

Defines

- `#define _GLIBCXX_BITSET_BITS_PER_WORD`
- `#define _GLIBCXX_BITSET_WORDS\(__n\)`

Functions

- `template<size_t _Nb>`
`bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`

6.34.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [bitset](#).

6.35 bitset File Reference

Classes

- class [std::__debug::bitset<_Nb>](#)
Class [std::bitset](#) with additional safety/checking/debug instrumentation.
- struct [std::hash<__debug::bitset<_Nb>>](#)
[std::hash](#) specialization for [bitset](#).

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<size_t _Nb>
bitset<_Nb> std::__debug::operator& (const bitset<_Nb> &__x, const
bitset<_Nb> &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>
std::basic_ostream<_CharT, _Traits> & std::__debug::operator<<
(std::basic_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>
std::basic_istream<_CharT, _Traits> & std::__debug::operator>>
(std::basic_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>
bitset<_Nb> std::__debug::operator^ (const bitset<_Nb> &__x, const
bitset<_Nb> &__y)`
- `template<size_t _Nb>
bitset<_Nb> std::__debug::operator| (const bitset<_Nb> &__x, const
bitset<_Nb> &__y)`

6.35.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/bitset](#).

6.36 bitset File Reference

Classes

- class `std::__profile::bitset<_Nb>`
Class `std::bitset` wrapper with performance instrumentation.
- struct `std::hash<__profile::bitset<_Nb>>`
`std::hash` specialization for `bitset`.

Namespaces

- namespace `std`
- namespace `std::__profile`

Functions

- `template<size_t _Nb>`
`bitset<_Nb> std::__profile::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream<_CharT, _Traits> & std::__profile::operator<< (std::basic_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream<_CharT, _Traits> & std::__profile::operator>> (std::basic_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>`
`bitset<_Nb> std::__profile::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`
`bitset<_Nb> std::__profile::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`

6.36.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/bitset](#).

6.37 boost_concept_check.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Defines

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

Functions

- `template<class _Tp >`
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`
`void __gnu_cxx::__function_requires ()`

6.37.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost_concept_check.h](#).

6.38 boost_sp_counted_base.h File Reference

Classes

- class [std::tr1::bad_weak_ptr](#)
Exception possibly thrown by [shared_ptr](#).

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

Functions

- void [std::tr1::__throw_bad_weak_ptr\(\)](#)

6.38.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost_sp_counted_base.h](#).

6.39 c++0x_warning.h File Reference

6.39.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [c++0x_warning.h](#).

6.40 c++allocator.h File Reference

Defines

- #define [__glibcxx_base_allocator](#)

6.40.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++allocator.h](#).

6.41 c++config.h File Reference

Namespaces

- namespace [std](#)

Defines

- #define `__GLIBCXX__`
- #define `__glibcxx_assert(_Condition)`
- #define `__N(msgid)`
- #define `_GLIBCXX_ATOMIC_BUILTINS_1`
- #define `_GLIBCXX_ATOMIC_BUILTINS_2`
- #define `_GLIBCXX_ATOMIC_BUILTINS_4`
- #define `_GLIBCXX_ATOMIC_BUILTINS_8`
- #define `_GLIBCXX_BEGIN_EXTERN_C`
- #define `_GLIBCXX_BEGIN_LDBL_NAMESPACE`
- #define `_GLIBCXX_BEGIN_NAMESPACE(X)`
- #define `_GLIBCXX_BEGIN_NESTED_NAMESPACE(X, Y)`
- #define `_GLIBCXX_CONST`
- #define `_GLIBCXX_DEPRECATED_ATTR`
- #define `_GLIBCXX_END_EXTERN_C`
- #define `_GLIBCXX_END_LDBL_NAMESPACE`
- #define `_GLIBCXX_END_NAMESPACE`
- #define `_GLIBCXX_END_NESTED_NAMESPACE`
- #define `_GLIBCXX_EXTERN_TEMPLATE`
- #define `_GLIBCXX_FAST_MATH`
- #define `_GLIBCXX_HAS_GTHREADS`
- #define `_GLIBCXX_HAVE_ACOSF`
- #define `_GLIBCXX_HAVE_ACOSL`
- #define `_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- #define `_GLIBCXX_HAVE_ASINF`
- #define `_GLIBCXX_HAVE_ASINL`
- #define `_GLIBCXX_HAVE_ATAN2F`
- #define `_GLIBCXX_HAVE_ATAN2L`

- `#define _GLIBCXX_HAVE_ATANF`
- `#define _GLIBCXX_HAVE_ATANL`
- `#define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- `#define _GLIBCXX_HAVE_CEILF`
- `#define _GLIBCXX_HAVE_CEILL`
- `#define _GLIBCXX_HAVE_COMPLEX_H`
- `#define _GLIBCXX_HAVE_COSF`
- `#define _GLIBCXX_HAVE_COSHF`
- `#define _GLIBCXX_HAVE_COSHL`
- `#define _GLIBCXX_HAVE_COSL`
- `#define _GLIBCXX_HAVE_DLFCN_H`
- `#define _GLIBCXX_HAVE_EBADMSG`
- `#define _GLIBCXX_HAVE_ECANCELED`
- `#define _GLIBCXX_HAVE_EIDRM`
- `#define _GLIBCXX_HAVE_ENDIAN_H`
- `#define _GLIBCXX_HAVE_ENODATA`
- `#define _GLIBCXX_HAVE_ENOLINK`
- `#define _GLIBCXX_HAVE_ENOSR`
- `#define _GLIBCXX_HAVE_ENOSTR`
- `#define _GLIBCXX_HAVE_ENOTRECOVERABLE`
- `#define _GLIBCXX_HAVE_ENOTSUP`
- `#define _GLIBCXX_HAVE_EOVERFLOW`
- `#define _GLIBCXX_HAVE_EOWNERDEAD`
- `#define _GLIBCXX_HAVE_EPROTO`
- `#define _GLIBCXX_HAVE_ETIME`
- `#define _GLIBCXX_HAVE_ETXTBSY`
- `#define _GLIBCXX_HAVE_EXECINFO_H`
- `#define _GLIBCXX_HAVE_EXPF`
- `#define _GLIBCXX_HAVE_EXPL`
- `#define _GLIBCXX_HAVE_FABSF`
- `#define _GLIBCXX_HAVE_FABSL`
- `#define _GLIBCXX_HAVE_FENV_H`
- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOAT_H`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_GETIPINFO`

- #define **_GLIBCXX_HAVE_GTHR_DEFAULT**
- #define **_GLIBCXX_HAVE_HYPOT**
- #define **_GLIBCXX_HAVE_HYPOTF**
- #define **_GLIBCXX_HAVE_HYPOTL**
- #define **_GLIBCXX_HAVE_ICONV**
- #define **_GLIBCXX_HAVE_INT64_T**
- #define **_GLIBCXX_HAVE_INT64_T_LONG**
- #define **_GLIBCXX_HAVE_INTTYPES_H**
- #define **_GLIBCXX_HAVE_ISINF**
- #define **_GLIBCXX_HAVE_ISINFF**
- #define **_GLIBCXX_HAVE_ISINFL**
- #define **_GLIBCXX_HAVE_ISNAN**
- #define **_GLIBCXX_HAVE_ISNANF**
- #define **_GLIBCXX_HAVE_ISNANL**
- #define **_GLIBCXX_HAVE_ISWBLANK**
- #define **_GLIBCXX_HAVE_LC_MESSAGES**
- #define **_GLIBCXX_HAVE_LDEXPF**
- #define **_GLIBCXX_HAVE_LDEXPL**
- #define **_GLIBCXX_HAVE_LIBINTL_H**
- #define **_GLIBCXX_HAVE_LIMIT_AS**
- #define **_GLIBCXX_HAVE_LIMIT_DATA**
- #define **_GLIBCXX_HAVE_LIMIT_FSIZE**
- #define **_GLIBCXX_HAVE_LIMIT_RSS**
- #define **_GLIBCXX_HAVE_LIMIT_VMEM**
- #define **_GLIBCXX_HAVE_LINUX_FUTEX**
- #define **_GLIBCXX_HAVE_LOCALE_H**
- #define **_GLIBCXX_HAVE_LOG10F**
- #define **_GLIBCXX_HAVE_LOG10L**
- #define **_GLIBCXX_HAVE_LOGF**
- #define **_GLIBCXX_HAVE_LOGL**
- #define **_GLIBCXX_HAVE_MBSTATE_T**
- #define **_GLIBCXX_HAVE_MEMORY_H**
- #define **_GLIBCXX_HAVE_MODF**
- #define **_GLIBCXX_HAVE_MODFF**
- #define **_GLIBCXX_HAVE_MODFL**
- #define **_GLIBCXX_HAVE_POLL**
- #define **_GLIBCXX_HAVE_POWF**
- #define **_GLIBCXX_HAVE_POWL**
- #define **_GLIBCXX_HAVE_S_ISREG**
- #define **_GLIBCXX_HAVE_SETENV**
- #define **_GLIBCXX_HAVE_SINCOS**
- #define **_GLIBCXX_HAVE_SINCOSF**
- #define **_GLIBCXX_HAVE_SINCOSL**

- #define **_GLIBCXX_HAVE_SINF**
- #define **_GLIBCXX_HAVE_SINHF**
- #define **_GLIBCXX_HAVE_SINHL**
- #define **_GLIBCXX_HAVE_SINL**
- #define **_GLIBCXX_HAVE_SQRTF**
- #define **_GLIBCXX_HAVE_SQRTL**
- #define **_GLIBCXX_HAVE_STDBOOL_H**
- #define **_GLIBCXX_HAVE_STDINT_H**
- #define **_GLIBCXX_HAVE_STDLIB_H**
- #define **_GLIBCXX_HAVE_STRERROR_L**
- #define **_GLIBCXX_HAVE_STRERROR_R**
- #define **_GLIBCXX_HAVE_STRING_H**
- #define **_GLIBCXX_HAVE_STRINGS_H**
- #define **_GLIBCXX_HAVE_STRTOF**
- #define **_GLIBCXX_HAVE_STRTOLD**
- #define **_GLIBCXX_HAVE_STRXFRM_L**
- #define **_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT**
- #define **_GLIBCXX_HAVE_SYS_IOCTL_H**
- #define **_GLIBCXX_HAVE_SYS_IPC_H**
- #define **_GLIBCXX_HAVE_SYS_PARAM_H**
- #define **_GLIBCXX_HAVE_SYS_RESOURCE_H**
- #define **_GLIBCXX_HAVE_SYS_SEM_H**
- #define **_GLIBCXX_HAVE_SYS_STAT_H**
- #define **_GLIBCXX_HAVE_SYS_TIME_H**
- #define **_GLIBCXX_HAVE_SYS_TYPES_H**
- #define **_GLIBCXX_HAVE_SYS_UIO_H**
- #define **_GLIBCXX_HAVE_TANF**
- #define **_GLIBCXX_HAVE_TANHF**
- #define **_GLIBCXX_HAVE_TANHL**
- #define **_GLIBCXX_HAVE_TANL**
- #define **_GLIBCXX_HAVE_TGMATH_H**
- #define **_GLIBCXX_HAVE_TLS**
- #define **_GLIBCXX_HAVE_UNISTD_H**
- #define **_GLIBCXX_HAVE_VFWSCANF**
- #define **_GLIBCXX_HAVE_VSWSCANF**
- #define **_GLIBCXX_HAVE_VWSCANF**
- #define **_GLIBCXX_HAVE_WCHAR_H**
- #define **_GLIBCXX_HAVE_WCSTOF**
- #define **_GLIBCXX_HAVE_WCTYPE_H**
- #define **_GLIBCXX_HAVE_WRITEV**
- #define **_GLIBCXX_HOSTED**
- #define **_GLIBCXX_ICONV_CONST**

- `#define _GLIBCXX_LDBL_NAMESPACE`
- `#define _GLIBCXX_NAMESPACE_ASSOCIATION_VERSION`
- `#define _GLIBCXX_NORETURN`
- `#define _GLIBCXX_NOTHROW`
- `#define _GLIBCXX_PACKAGE__GLIBCXX_VERSION`
- `#define _GLIBCXX_PACKAGE_BUGREPORT`
- `#define _GLIBCXX_PACKAGE_NAME`
- `#define _GLIBCXX_PACKAGE_STRING`
- `#define _GLIBCXX_PACKAGE_TARNAME`
- `#define _GLIBCXX_PACKAGE_URL`
- `#define _GLIBCXX_PSEUDO_VISIBILITY(V)`
- `#define _GLIBCXX_PURE`
- `#define _GLIBCXX_RES_LIMITS`
- `#define _GLIBCXX_STD`
- `#define _GLIBCXX_STD_D`
- `#define _GLIBCXX_STD_P`
- `#define _GLIBCXX_STD_PR`
- `#define _GLIBCXX_STDIO_MACROS`
- `#define _GLIBCXX_SYMVER`
- `#define _GLIBCXX_SYMVER_GNU`
- `#define _GLIBCXX_USE_C99`
- `#define _GLIBCXX_USE_C99_COMPLEX`
- `#define _GLIBCXX_USE_C99_COMPLEX_TR1`
- `#define _GLIBCXX_USE_C99_CTYPE_TR1`
- `#define _GLIBCXX_USE_C99_FENV_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- `#define _GLIBCXX_USE_C99_MATH`
- `#define _GLIBCXX_USE_C99_MATH_TR1`
- `#define _GLIBCXX_USE_C99_STDINT_TR1`
- `#define _GLIBCXX_USE_DECIMAL_FLOAT`
- `#define _GLIBCXX_USE_GETTIMEOFDAY`
- `#define _GLIBCXX_USE_LFS`
- `#define _GLIBCXX_USE_LONG_LONG`
- `#define _GLIBCXX_USE_NLS`
- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_VISIBILITY_ATTR(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define LT_OBJDIR`
- `#define STDC_HEADERS`

Typedefs

- typedef __PTRDIFF_TYPE__ **std::ptrdiff_t**
- typedef __SIZE_TYPE__ **std::size_t**

Functions

- typedef **std::decltype** (nullptr) nullptr_t

6.41.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++config.h](#).

6.42 c++io.h File Reference

Namespaces

- namespace [std](#)

Typedefs

- typedef FILE **std::__c_file**
- typedef __thread_mutex_t **std::__c_lock**

6.42.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++io.h](#).

6.43 c++locale.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _GLIBCXX_C_LOCALE_GNU`
- `#define _GLIBCXX_NUM_CATEGORIES`

Typedefs

- `typedef __locale_t std::__c_locale`

Functions

- `int std::__convert_from_v (const __c_locale &__cloc __attribute__((__unused__)), char *__out, const int __size __attribute__((__unused__)), const char *__fmt,...)`

6.43.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale.h](#).

6.44 c++locale_internal.h File Reference

6.44.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale_internal.h](#).

6.45 cassert File Reference

6.45.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cassert](#).

6.46 ccomplex File Reference

6.46.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ccomplex](#).

6.47 ccomplex File Reference

6.47.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ccomplex](#).

6.48 ctype File Reference

Namespaces

- namespace [std](#)

6.48.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ctype](#).

6.49 ctype File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`

- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.49.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cctype](#).

6.50 cctype File Reference

6.50.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cctype](#).

6.51 cerrno File Reference

Defines

- `#define errno`

6.51.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cerrno](#).

6.52 cenv File Reference

6.52.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cfenv](#).

6.53 `cfenv` File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.53.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfenv](#).

6.54 `cfenv` File Reference

6.54.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cfenv](#).

6.55 `cfloat` File Reference

Defines

- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

6.55.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cfloat](#).

6.56 cfloat File Reference

Defines

- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

6.56.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfloat](#).

6.57 char_traits.h File Reference

Classes

- struct [__gnu_cxx::_Char_types<_CharT>](#)
Mapping from character type to associated types.
- struct [__gnu_cxx::char_traits<_CharT>](#)
Base class used to implement [std::char_traits](#).
- struct [std::char_traits<_CharT>](#)
Basis for explicit traits specializations.
- struct [std::char_traits<char>](#)
21.1.3.1 [char_traits](#) specializations
- struct [std::char_traits<wchar_t>](#)
21.1.3.2 [char_traits](#) specializations

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Defines

- `#define _CHAR_TRAITS_EOF`

6.57.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [char_traits.h](#).

6.58 checkers.h File Reference

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _Compare >
bool __gnu_parallel::__is_sorted (_Iter __begin, _Iter __end, _Compare __-
comp)`

6.58.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [checkers.h](#).

6.59 chrono File Reference

Classes

- `struct std::chrono::duration< _Rep, _Period >
duration`

- struct `std::chrono::duration_values< _Rep >`
duration_values
- struct `std::chrono::system_clock`
system_clock
- struct `std::chrono::time_point< _Clock, _Duration >`
time_point
- struct `std::chrono::treat_as_floating_point< _Rep >`
treat_as_floating_point

Namespaces

- namespace `std`
- namespace `std::chrono`

Typedefs

- typedef system_clock **`std::chrono::high_resolution_clock`**
- typedef duration< int, ratio< 3600 > > `std::chrono::hours`
- typedef duration< int64_t, micro > `std::chrono::microseconds`
- typedef duration< int64_t, milli > `std::chrono::milliseconds`
- typedef duration< int, ratio< 60 > > `std::chrono::minutes`
- typedef system_clock **`std::chrono::monotonic_clock`**
- typedef duration< int64_t, nano > `std::chrono::nanoseconds`
- typedef duration< int64_t > `std::chrono::seconds`

Functions

- template<typename _ToDuration , typename _Rep , typename _Period >
enable_if< __is_duration< _ToDuration >::value, _ToDuration >::type
`std::chrono::duration_cast` (const duration< _Rep, _Period > &__d)
- template<typename _Clock , typename _Duration1 , typename _Duration2 >
bool **`std::chrono::operator!=`** (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
bool **`std::chrono::operator!=`** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type std::chrono::operator% (const duration< _Rep1, _Period1 > &__lhs,`
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, typename enable_`
`if<!__is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period >`
`std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _`
`Rep2 &__s)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, _Rep2 >::type, _Period >`
`std::chrono::operator* (const duration< _Rep1, _Period > &__d, const _Rep2`
`&__s)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep2, _Rep1 >::type, _Period >`
`std::chrono::operator* (const _Rep1 &__s, const duration< _Rep2, _Period >`
`&__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs,`
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >`
`time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`
`_Period2 > >::type > std::chrono::operator+ (const time_point< _Clock, _`
`Duration1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Duration2 >`
`time_point< _Clock, typename common_type< duration< _Rep1, _Period1`
`>, _Duration2 >::type > std::chrono::operator+ (const duration< _Rep1, _`
`Period1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >`
`time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,`
`_Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _`
`Duration1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`common_type< _Duration1, _Duration2 >::type std::chrono::operator- (const`
`time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _`
`Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs,`
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, typename enable_`
`if<!__is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period >`

std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
common_type< _Rep1, _Rep2 >::type **std::chrono::operator/** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Duration2 >
bool **std::chrono::operator<** (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool **std::chrono::operator<** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool **std::chrono::operator<=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Duration2 >
bool **std::chrono::operator<=** (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Duration2 >
bool **std::chrono::operator==** (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool **std::chrono::operator==** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Duration2 >
bool **std::chrono::operator>** (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool **std::chrono::operator>** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Duration2 >
bool **std::chrono::operator>=** (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool **std::chrono::operator>=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _ToDuration, typename _Clock, typename _Duration >
enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _ToDuration > >::type **std::chrono::time_point_cast** (const time_point< _Clock, _Duration > &__t)

6.59.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [chrono](#).

6.60 `cinttypes` File Reference

6.60.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cinttypes](#).

6.61 `cinttypes` File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.61.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cinttypes](#).

6.62 `cinttypes` File Reference

6.62.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cinttypes](#).

6.63 `ciso646` File Reference

6.63.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ciso646](#).

6.64 `climits` File Reference

Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

6.64.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [climits](#).

6.65 `climits` File Reference

Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

6.65.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/climits](#).

6.66 clocale File Reference

Namespaces

- namespace [std](#)

6.66.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [clocale](#).

6.67 cmath File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp >
_Tp std::__cmath_power (_Tp, unsigned int)`
- `template<typename _Tp >
_Tp std::__pow_helper (_Tp __x, int __n)`
- `float std::abs (float __x)`
- `long double std::abs (long double __x)`
- `double std::abs (double __x)`
- `template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::abs (_Tp __x)`
- `long double std::acos (long double __x)`
- `template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::acos (_Tp __x)`
- `float std::acos (float __x)`
- `float std::asin (float __x)`
- `long double std::asin (long double __x)`

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::asin (_Tp __x)`
- `float std::atan (float __x)`
- `long double std::atan (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::atan (_Tp __x)`
- `float std::atan2 (float __y, float __x)`
- `long double std::atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`
`>::__type, _Up >::__type std::atan2 (_Tp __y, _Up __x)`
- `float std::ceil (float __x)`
- `long double std::ceil (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::ceil (_Tp __x)`
- `float std::cos (float __x)`
- `long double std::cos (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::cos (_Tp __x)`
- `float std::cosh (float __x)`
- `long double std::cosh (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::cosh (_Tp __x)`
- `float std::exp (float __x)`
- `long double std::exp (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::exp (_Tp __x)`
- `long double std::fabs (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::fabs (_Tp __x)`
- `float std::fabs (float __x)`
- `float std::floor (float __x)`
- `long double std::floor (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::floor (_Tp __x)`

- float **std::fmod** (float __x, float __y)
- long double **std::fmod** (long double __x, long double __y)
- long double **std::frexp** (long double __x, int *__exp)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::frexp (_Tp __x, int *__exp)
- float **std::frexp** (float __x, int *__exp)
- float **std::ldexp** (float __x, int __exp)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::ldexp (_Tp __x, int __exp)
- long double **std::ldexp** (long double __x, int __exp)
- float **std::log** (float __x)
- long double **std::log** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::log (_Tp __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::log10 (_Tp __x)
- float **std::log10** (float __x)
- long double **std::log10** (long double __x)
- long double **std::modf** (long double __x, long double *__iptr)
- float **std::modf** (float __x, float *__iptr)
- long double **std::pow** (long double __x, long double __y)
- template<typename _Tp, typename _Up >
__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value, _Tp >::__type, _Up >::__type **std::pow** (_Tp __x, _Up __y)
- float **std::pow** (float __x, float __y)
- long double **std::sin** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::sin (_Tp __x)
- float **std::sin** (float __x)
- long double **std::sinh** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
std::sinh (_Tp __x)
- float **std::sinh** (float __x)
- float **std::sqrt** (float __x)
- long double **std::sqrt** (long double __x)

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::sqrt (_Tp __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::tan (_Tp __x)`
- `long double std::tan (long double __x)`
- `float std::tan (float __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`std::tanh (_Tp __x)`
- `long double std::tanh (long double __x)`
- `float std::tanh (float __x)`

6.67.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cmath](#).

6.68 cmath File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

Functions

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int`
`__n, unsigned int __m, _Tp __x)`

- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::assoc_legendre** (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx, typename _Tpy >
__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type **std::tr1::beta** (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::comp_ellint_1** (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::comp_ellint_2** (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn >
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type **std::tr1::comp_ellint_3** (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type **std::tr1::conf_hyperg** (_Tpa __a, _Tpc __c, _Tp __x)
- float **std::tr1::conf_hypergf** (float __a, float __c, float __x)
- long double **std::tr1::conf_hypergl** (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **std::tr1::cyl_bessel_i** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **std::tr1::cyl_bessel_j** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)

- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_-`
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl_bessel_kf (float __nu, float __x)`
- `long double std::tr1::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_-`
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl_neumannf (float __nu, float __x)`
- `long double std::tr1::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _-`
`Tpp __phi)`
- `float std::tr1::ellint_1f (float __k, float __phi)`
- `long double std::tr1::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _-`
`Tpp __phi)`
- `float std::tr1::ellint_2f (float __k, float __phi)`
- `long double std::tr1::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp`
`__k, _Tpn __nu, _Tpp __phi)`
- `float std::tr1::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::tr1::ellint_3l (long double __k, long double __nu, long double`
`__phi)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)`
- `float std::tr1::expintf (float __x)`
- `long double std::tr1::expintl (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp`
`__x)`
- `float std::tr1::hermitef (unsigned int __n, float __x)`
- `long double std::tr1::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg`
`(_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`
- `long double std::tr1::hypergl (long double __a, long double __b, long double`
`__c, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _-`
`Tp __x)`

- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::legendre** (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- float **std::tr1::pow** (float __x, float __y)
- template<typename _Tp, typename _Up >
__gnu_cxx::__promote_2< _Tp, _Up >::__type **std::tr1::pow** (_Tp __x, _Up __y)
- double **std::tr1::pow** (double __x, double __y)
- long double **std::tr1::pow** (long double __x, long double __y)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::riemann_zeta** (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_bessel** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_legendre** (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_neumann** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

6.68.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cmath](#).

6.69 cmath File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

6.69.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cmath](#).

6.70 cmath.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp >
_Tp std::__cmath_power (_Tp, unsigned int)`

6.70.1 Detailed Description

This is a Standard C++ Library file.

Definition in file [cmath.tcc](#).

6.71 codecvt.h File Reference

Classes

- class [std::__codecvt_abstract_base](#)< [_InternT](#), [_ExternT](#), [_StateT](#) >
Common base for codecvt functions.
- class [std::codecvt](#)< [_InternT](#), [_ExternT](#), [_StateT](#) >

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

- class `std::codecvt< char, char, mbstate_t >`
class `codecvt<char, char, mbstate_t>` specialization.
- class `std::codecvt< wchar_t, char, mbstate_t >`
class `codecvt<wchar_t, char, mbstate_t>` specialization.
- class `std::codecvt_base`
Empty base class for `codecvt` facet [22.2.1.5].
- class `std::codecvt_byname< _InternT, _ExternT, _StateT >`
class `codecvt_byname` [22.2.1.6].

Namespaces

- namespace `std`

6.71.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `codecvt.h`.

6.72 codecvt_specializations.h File Reference

Classes

- struct `__gnu_cxx::encoding_char_traits< _CharT >`
`encoding_char_traits`
- class `__gnu_cxx::encoding_state`
Extension to use `iconv` for dealing with character encodings.
- class `std::codecvt< _InternT, _ExternT, encoding_state >`
`codecvt<InternT, _ExternT, encoding_state>` specialization.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<typename _Tp >
size_t std::__iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **,
size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf,
size_t *__outbytes)`

6.72.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [codecvt_specializations.h](#).

6.73 compatibility.h File Reference

6.73.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

Definition in file [x86_64-unknown-linux-gnu/bits/compatibility.h](#).

6.74 compatibility.h File Reference

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Tp >
bool __gnu_parallel::__compare_and_swap (volatile _Tp *__ptr, _Tp __-
comparand, _Tp __replacement)`

- `bool __gnu_parallel::__compare_and_swap_32` (volatile `int32_t *``__ptr`, `int32_t` `__comparand`, `int32_t` `__replacement`)
- `bool __gnu_parallel::__compare_and_swap_64` (volatile `int64_t *``__ptr`, `int64_t` `__comparand`, `int64_t` `__replacement`)
- `template<typename _Tp >`
`_Tp __gnu_parallel::__fetch_and_add` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- `int32_t __gnu_parallel::__fetch_and_add_32` (volatile `int32_t *``__ptr`, `int32_t` `__addend`)
- `int64_t __gnu_parallel::__fetch_and_add_64` (volatile `int64_t *``__ptr`, `int64_t` `__addend`)
- `void __gnu_parallel::__yield` ()

6.74.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/compatibility.h](#).

6.75 `compiletime_settings.h` File Reference

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Defines

- `#define _GLIBCXX_ASSERTIONS`
- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

6.75.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [compiletime_settings.h](#).

6.75.2 Define Documentation

6.75.2.1 `#define _GLIBCXX_ASSERTIONS`

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

6.75.2.2 `#define _GLIBCXX_CALL(__n)`

Macro to produce log message when entering a function.

Parameters

`__n` Input size.

See also

[`_GLIBCXX_VERBOSE_LEVEL`](#)

Definition at line 44 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge_loser_tree()`, `__gnu_parallel::__multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::__multiway_merge_loser_tree_unguarded()`, `__gnu_parallel::__multiway_merge_sentinels()`, `__gnu_parallel::__parallel_multiway_merge()`, and `__gnu_parallel::__parallel_sort_mwms()`.

6.75.2.3 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the L1 cache for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `compiletime_settings.h`.

6.75.2.4 #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 74 of file `comPILEtime_settings.h`.

6.75.2.5 #define _GLIBCXX_SCALE_DOWN_FPU

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `comPILEtime_settings.h`.

6.75.2.6 #define _GLIBCXX_VERBOSE_LEVEL

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `comPILEtime_settings.h`.

6.76 complex File Reference**Classes**

- struct [std::complex<_Tp>](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<typename _Tp>`
`_Tp std::__complex_abs (const complex<_Tp> &__z)`
- `template<typename _Tp>`
`_Tp std::__complex_arg (const complex<_Tp> &__z)`
- `template<typename _Tp>`
`complex<_Tp> std::__complex_cos (const complex<_Tp> &__z)`
- `template<typename _Tp>`
`complex<_Tp> std::__complex_cosh (const complex<_Tp> &__z)`

- `template<typename _Tp >`
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const`
`complex< _Tp > &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__-`
`z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::conj (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`

- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::proj (_Tp __x)`
- `template<typename _Tp >`
`_Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tanh (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const`
`complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp`
`&__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp >`
`&__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const`
`complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp`
`&__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp >`
`&__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const`
`complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp`
`&__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp >`
`&__y)`

- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const complex< _Tp >`
`&__y)`
- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp >`
`&__y)`
- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

6.76.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex](#).

6.77 complex File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

Functions

- `template<typename _Tp >
std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >
std::complex< typename __gnu_cxx::__promote< _Tp >::__type >
std::tr1::conj (_Tp __x)`
- `template<typename _Tp, typename _Up >
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >
std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const
_Tp &__y)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp
> &__y)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const
std::complex< _Tp > &__y)`

6.77.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/complex](#).

6.78 complex File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

Functions

- `template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_acos (const std::complex< _Tp >
&__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_acosh (const std::complex< _Tp >
&__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_asin (const std::complex< _Tp >
&__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_asinh (const std::complex< _Tp >
&__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_atan (const std::complex< _Tp >
&__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_atanh (const std::complex< _Tp >
&__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type std::tr1::arg (_Tp __x)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >
std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`

- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::imag (_Tp)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::norm (_Tp __x)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up >`
`&__y)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::real (_Tp __x)`

6.78.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/complex](#).

6.79 complex.h File Reference

6.79.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex.h](#).

6.80 `concept_check.h` File Reference

Defines

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

6.80.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concept_check.h](#).

6.81 `concurrency.h` File Reference

Classes

- class [__gnu_cxx::__scoped_lock](#)
Scoped lock idiom.

Namespaces

- namespace [__gnu_cxx](#)

Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

6.81.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concurrency.h](#).

6.82 cond_dealtor.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- `#define PB_DS_COND DEALTOR_CLASS_C_DEC`
- `#define PB_DS_COND DEALTOR_CLASS_T_DEC`

6.82.1 Detailed Description

Contains a conditional deallocator.

Definition in file [cond_dealtor.hpp](#).

6.83 condition_variable File Reference

Classes

- class [std::condition_variable](#)
condition_variable
- class [std::condition_variable_any](#)
condition_variable_any

Namespaces

- namespace [std](#)

Enumerations

- enum [std::cv_status](#) { **no_timeout**, **timeout** }

6.83.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [condition_variable](#).

6.84 constructors_destructor_fn_imps.hpp File Reference

Functions

- **PB_DS_CLASS_NAME** ()
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 >
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 >
PB_DS_CLASS_NAME (T0 t0, T1 t1)

- `template<typename T0 >`
`PB_DS_CLASS_NAME` (T0 t0)
- `PB_DS_CLASS_NAME` (const PB_DS_CLASS_NAME &other)

6.84.1 Detailed Description

Contains `constructors_destructor_fn_imps` applicable to different containers.

Definition in file [constructors_destructor_fn_imps.hpp](#).

6.85 container_base_dispatch.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.85.1 Detailed Description

Contains an associative container dispatching base.

Definition in file [container_base_dispatch.hpp](#).

6.86 cpp_type_traits.h File Reference

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

6.86.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpp_type_traits.h](#).

6.87 `cpu_defines.h` File Reference

6.87.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpu_defines.h](#).

6.88 `csetjmp` File Reference

Namespaces

- namespace [std](#)

Defines

- #define `setjmp(env)`

6.88.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csetjmp](#).

6.89 `csignal` File Reference

Namespaces

- namespace [std](#)

6.89.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csignal](#).

6.90 cstdarg File Reference

Namespaces

- namespace [std](#)

Defines

- #define `va_end`(ap)

6.90.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdarg](#).

6.91 cstdarg File Reference

6.91.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdarg](#).

6.92 cstdbool File Reference

6.92.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdbool](#).

6.93 `cstdbool` File Reference

6.93.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdbool](#).

6.94 `cstddef` File Reference

6.94.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstddef](#).

6.95 `cstdint` File Reference

6.95.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdint](#).

6.96 `cstdint` File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.96.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

6.97 cstdint File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

6.97.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cstdint](#).

6.98 cstdio File Reference

Namespaces

- namespace [std](#)

6.98.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdio](#).

6.99 cstdio File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.99.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdio](#).

6.100 cstdio File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

6.100.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cstdio](#).

6.101 cstdlib File Reference

Namespaces

- namespace [std](#)

Defines

- `#define EXIT_FAILURE`
- `#define EXIT_SUCCESS`

Functions

- void **std::abort** (void) _GLIBCXX_NORETURN throw ()
- int **std::atexit** (void(*)()) throw ()
- void **std::exit** (int) _GLIBCXX_NORETURN throw ()

6.101.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdlib](#).

6.102 cstdlib File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.102.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdlib](#).

6.103 cstdlib File Reference

6.103.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cstdlib](#).

6.104 cstring File Reference

Namespaces

- namespace [std](#)

Functions

- void * **std::memchr** (void *__s, int __c, size_t __n)
- char * **std::strchr** (char *__s, int __n)
- char * **std::strpbrk** (char *__s1, const char *__s2)
- char * **std::strrchr** (char *__s, int __n)
- char * **std::strstr** (char *__s1, const char *__s2)

6.104.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstring](#).

6.105 ctgmath File Reference

6.105.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ctgmath](#).

6.106 ctgmath File Reference

6.106.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctgmath](#).

6.107 ctime File Reference

Namespaces

- namespace [std](#)

6.107.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ctime](#).

6.108 ctime File Reference

6.108.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctime](#).

6.109 ctype_base.h File Reference

Classes

- struct [std::ctype_base](#)

Base class for ctype.

Namespaces

- namespace [std](#)

6.109.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype_base.h](#).

6.110 ctype_inline.h File Reference

Namespaces

- namespace [std](#)

6.110.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype_inline.h](#).

6.111 ctype_noninline.h File Reference

6.111.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype_noninline.h](#).

6.112 cwchar File Reference

Namespaces

- namespace [std](#)

Functions

- `wchar_t * std::wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcpbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`

6.112.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwchar](#).

6.113 `wchar` File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.113.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwchar](#).

6.114 `wchar` File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

6.114.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cwchar](#).

6.115 `cwctype` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _GLIBCXX_CWCTYPE`

6.115.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwctype](#).

6.116 cwctype File Reference

Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

6.116.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwctype](#).

6.117 cwctype File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

6.117.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/cwctype](#).

6.118 cxxabi-forced.h File Reference

Classes

- class [__cxxabiv1::__forced_unwind](#)

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

6.118.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi-forced.h](#).

6.119 cxxabi.h File Reference

Classes

- class [__gnu_cxx::recursive_init_error](#)

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [abi](#)

Defines

- `#define _GLIBCXX_NOTHROW`

Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type)(void *)`

Functions

- `int __cxxabiv1::__cxa_atexit (void(*)(void *), void *, void *) throw ()`
- `void __cxxabiv1::__cxa_bad_cast ()`
- `void __cxxabiv1::__cxa_bad_typeid ()`
- `std::type_info * __cxxabiv1::__cxa_current_exception_type () __attribute__((__pure__)) throw ()`
- `char * __cxxabiv1::__cxa_demangle (const char *__mangled_name, char *__output_buffer, size_t *__length, int *__status)`
- `int __cxxabiv1::__cxa_finalize (void *)`
- `void __cxxabiv1::__cxa_guard_abort (__guard *) throw ()`
- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *) throw ()`
- `void __cxxabiv1::__cxa_pure_virtual (void) __attribute__((__noreturn__))`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void *dest_array, void *src_array, size_t element_count, size_t element_size, __cxa_ctor_return_type(*constructor)(void *, void *), __cxa_ctor_type destructor)`
- `void __cxxabiv1::__cxa_vec_cleanup (void *__array_address, size_t __element_count, size_t __s, __cxa_ctor_type destructor) throw ()`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void *__array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type constructor, __cxa_ctor_type destructor)`
- `void __cxxabiv1::__cxa_vec_delete (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type destructor)`
- `void __cxxabiv1::__cxa_vec_delete2 (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type destructor, void(*__dealloc)(void *))`
- `void __cxxabiv1::__cxa_vec_delete3 (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type destructor, void(*__dealloc)(void *, size_t))`
- `void __cxxabiv1::__cxa_vec_dtor (void *__array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type destructor)`
- `void * __cxxabiv1::__cxa_vec_new (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type constructor, __cxa_ctor_type destructor)`
- `void * __cxxabiv1::__cxa_vec_new2 (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type constructor, __cxa_ctor_type destructor, void *(*__alloc)(size_t), void(*__dealloc)(void *))`
- `void * __cxxabiv1::__cxa_vec_new3 (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type constructor, __cxa_ctor_type destructor, void *(*__alloc)(size_t), void(*__dealloc)(void *, size_t))`
- `void * __cxxabiv1::__dynamic_cast (const void *__src_ptr, const __class_type_info *__src_type, const __class_type_info *__dst_type, ptrdiff_t __src2dst)`

6.119.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi.h](#).

6.120 cxxabi_tweaks.h File Reference

Defines

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

Typedefs

- `typedef void __cxxabiv1::__cxa_cdtor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI_ - __)))`

6.120.1 Detailed Description

The header provides an CPU-variable interface to the C++ ABI.

Definition in file [cxxabi_tweaks.h](#).

6.121 debug.h File Reference

Namespaces

- namespace [__gnu_debug](#)
- namespace [std](#)
- namespace [std::__debug](#)

Defines

- #define `__glibcxx_requires_cond`(_Cond, _Msg)
- #define `__glibcxx_requires_heap`(_First, _Last)
- #define `__glibcxx_requires_heap_pred`(_First, _Last, _Pred)
- #define `__glibcxx_requires_nonempty`()
- #define `__glibcxx_requires_partitioned_lower`(_First, _Last, _Value)
- #define `__glibcxx_requires_partitioned_lower_pred`(_First, _Last, _Value, _Pred)
- #define `__glibcxx_requires_partitioned_upper`(_First, _Last, _Value)
- #define `__glibcxx_requires_partitioned_upper_pred`(_First, _Last, _Value, _Pred)
- #define `__glibcxx_requires_sorted`(_First, _Last)
- #define `__glibcxx_requires_sorted_pred`(_First, _Last, _Pred)
- #define `__glibcxx_requires_sorted_set`(_First1, _Last1, _First2)
- #define `__glibcxx_requires_sorted_set_pred`(_First1, _Last1, _First2, _Pred)
- #define `__glibcxx_requires_string`(_String)
- #define `__glibcxx_requires_string_len`(_String, _Len)
- #define `__glibcxx_requires_subscript`(_N)
- #define `__glibcxx_requires_valid_range`(_First, _Last)
- #define `_GLIBCXX_DEBUG_ASSERT`(_Condition)
- #define `_GLIBCXX_DEBUG_ONLY`(_Statement)
- #define `_GLIBCXX_DEBUG_PEDASSERT`(_Condition)

6.121.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug.h](#).

6.122 `debug_allocator.h` File Reference

Classes

- class `__gnu_cxx::debug_allocator< _Alloc >`
*A meta-allocator with debugging bits, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls operator new
 - all deallocation calls operator delete.

Namespaces

- namespace [__gnu_cxx](#)

6.122.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. You should only include this header if you are using GCC 3 or later.

Definition in file [debug_allocator.h](#).

6.123 debug_map_base.hpp File Reference

6.123.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file [debug_map_base.hpp](#).

6.124 decimal File Reference

Classes

- class [std::decimal::decimal128](#)
3.2.4 Class [decimal128](#).
- class [std::decimal::decimal32](#)
3.2.2 Class [decimal32](#).
- class [std::decimal::decimal64](#)
3.2.3 Class [decimal64](#).

Namespaces

- namespace [std](#)
- namespace [std::decimal](#)

Defines

- #define **_DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_DEC**(_Op, _T1, _T2, _T3)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_INT**(_Op, _Tp)
- #define **_DECLARE_DECIMAL_COMPARISON**(_Op, _Tp)
- #define **_GLIBCXX_USE_DECIMAL_**

Functions

- double **std::decimal::decimal128_to_double** (decimal128 __d)
- float **std::decimal::decimal128_to_float** (decimal128 __d)
- long double **std::decimal::decimal128_to_long_double** (decimal128 __d)
- long long **std::decimal::decimal128_to_long_long** (decimal128 __d)
- double **std::decimal::decimal32_to_double** (decimal32 __d)
- float **std::decimal::decimal32_to_float** (decimal32 __d)
- long double **std::decimal::decimal32_to_long_double** (decimal32 __d)
- long long **std::decimal::decimal32_to_long_long** (decimal32 __d)
- double **std::decimal::decimal64_to_double** (decimal64 __d)
- float **std::decimal::decimal64_to_float** (decimal64 __d)
- long double **std::decimal::decimal64_to_long_double** (decimal64 __d)
- long long **std::decimal::decimal64_to_long_long** (decimal64 __d)
- double **std::decimal::decimal_to_double** (decimal32 __d)
- double **std::decimal::decimal_to_double** (decimal64 __d)
- double **std::decimal::decimal_to_double** (decimal128 __d)
- float **std::decimal::decimal_to_float** (decimal32 __d)
- float **std::decimal::decimal_to_float** (decimal64 __d)
- float **std::decimal::decimal_to_float** (decimal128 __d)
- long double **std::decimal::decimal_to_long_double** (decimal32 __d)
- long double **std::decimal::decimal_to_long_double** (decimal64 __d)
- long double **std::decimal::decimal_to_long_double** (decimal128 __d)
- long long **std::decimal::decimal_to_long_long** (decimal32 __d)
- long long **std::decimal::decimal_to_long_long** (decimal64 __d)
- long long **std::decimal::decimal_to_long_long** (decimal128 __d)
- static decimal128 **std::decimal::make_decimal128** (long long __coeff, int __exp)
- static decimal128 **std::decimal::make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (long long __coeff, int __exp)

- static decimal32 **std::decimal::make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (unsigned long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (long long __coeff, int __exp)

- bool **std::decimal::operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long __rhs)

- `bool std::decimal::operator!= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator!= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator!= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned long long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned int __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, int __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, long long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned long long __-
rhs)`
- `decimal32 std::decimal::operator* (int __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned int __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (long long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned long long __lhs, decimal32 __-
rhs)`
- `decimal64 std::decimal::operator* (decimal32 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, int __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned int __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, long long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned long long __-
rhs)`
- `decimal64 std::decimal::operator* (int __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned int __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (long long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned long long __lhs, decimal64 __-
rhs)`
- `decimal128 std::decimal::operator* (decimal32 __lhs, decimal128 __rhs)`
- `decimal128 std::decimal::operator* (decimal64 __lhs, decimal128 __rhs)`
- `decimal128 std::decimal::operator* (decimal128 __lhs, decimal32 __rhs)`
- `decimal128 std::decimal::operator* (decimal128 __lhs, decimal64 __rhs)`

- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (long long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator+** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, int __rhs)
- decimal128 **std::decimal::operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long __rhs)
- decimal128 **std::decimal::operator+** (unsigned long long __lhs, decimal128 __rhs)

- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator+** (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (long long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator+** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __rhs)
- decimal64 **std::decimal::operator+** (long long __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator-** (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (long long __lhs, decimal32 __rhs)

- decimal32 **std::decimal::operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator-** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator-** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (long long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long __rhs)

- decimal32 **std::decimal::operator/** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator/** (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator/** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (long long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator/** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (long __lhs, decimal128 __rhs)

- decimal128 **std::decimal::operator/** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal64 __rhs)

- `bool std::decimal::operator< (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned long __rhs)`

- `bool std::decimal::operator==(decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator==(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator==(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, long long __rhs)`

- `bool std::decimal::operator> (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator> (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator> (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator> (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator> (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal128 __rhs)`

- bool **std::decimal::operator>=** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal64 __rhs)

6.124.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [decimal](#).

6.125 deque File Reference

6.125.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [deque](#).

6.126 deque File Reference

Classes

- class [std::__debug::deque< _Tp, _Allocator >](#)
Class [std::deque](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

6.126.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/deque](#).

6.127 deque File Reference

Classes

- class [std::__profile::deque< _Tp, _Allocator >](#)
Class [std::deque](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

6.127.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/deque](#).

6.128 deque.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _`
`Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp,`
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

6.128.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [deque.tcc](#).

6.129 `enc_filebuf.h` File Reference

Classes

- class [__gnu_cxx::enc_filebuf<_CharT>](#)
class [enc_filebuf](#).

Namespaces

- namespace [__gnu_cxx](#)

6.129.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [enc_filebuf.h](#).

6.130 `equally_split.h` File Reference

Namespaces

- namespace `__gnu_parallel`

Functions

- `template<typename _DifferenceType, typename _OutputIterator >
_OutputIterator __gnu_parallel::equally_split (_DifferenceType __n, _-
ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >
_DifferenceType __gnu_parallel::equally_split_point (_DifferenceType __n, _-
ThreadIndex __num_threads, _ThreadIndex __thread_no)`

6.130.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [equally_split.h](#).

6.131 `error_constants.h` File Reference

Namespaces

- namespace `std`

Enumerations

- `enum errc {
 address_family_not_supported, address_in_use, address_not_available,
 already_connected,
 argument_list_too_long, argument_out_of_domain, bad_address, bad_-
 file_descriptor,
 bad_message, broken_pipe, connection_aborted, connection_already_in_-
 progress,
 connection_refused, connection_reset, cross_device_link, destination_-
 address_required,
 device_or_resource_busy, directory_not_empty, executable_format_error,
 file_exists,`

```

file_too_large,    filename_too_long,    function_not_supported,    host_ -
unreachable,

identifier_removed,    illegal_byte_sequence,    inappropriate_io_control_ -
operation, interrupted,

invalid_argument, invalid_seek, io_error, is_a_directory,

message_size, network_down, network_reset, network_unreachable,

no_buffer_space, no_child_process, no_link, no_lock_available,

no_message_available, no_message, no_protocol_option, no_space_on_ -
device,

no_stream_resources, no_such_device_or_address, no_such_device, no_ -
such_file_or_directory,

no_such_process, not_a_directory, not_a_socket, not_a_stream,

not_connected, not_enough_memory, not_supported, operation_canceled,

operation_in_progress,    operation_not_permitted,    operation_not_ -
supported, operation_would_block,

owner_dead, permission_denied, protocol_error, protocol_not_supported,

read_only_file_system,    resource_deadlock_would_occur,    resource_ -
unavailable_try_again, result_out_of_range,

state_not_recoverable, stream_timeout, text_file_busy, timed_out,

too_many_files_open_in_system,    too_many_files_open,    too_many_links,
too_many_symbolic_link_levels,

value_too_large, wrong_protocol_type }

```

6.131.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [error_constants.h](#).

6.132 exception File Reference

Classes

- class [std::bad_exception](#)
- class [std::exception](#)

Base class for all library exceptions.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Typedefs

- typedef void(* [std::terminate_handler](#))()
- typedef void(* [std::unexpected_handler](#))()

Functions

- void [__gnu_cxx::__verbose_terminate_handler](#) ()
- terminate_handler [std::set_terminate](#) (terminate_handler) throw ()
- unexpected_handler [std::set_unexpected](#) (unexpected_handler) throw ()
- void [std::terminate](#) () [__attribute__\(\(__noreturn__\)\)](#) throw ()
- bool [std::uncaught_exception](#) () [__attribute__\(\(__pure__\)\)](#) throw ()
- void [std::unexpected](#) () [__attribute__\(\(__noreturn__\)\)](#)

6.132.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [exception](#).

6.133 exception.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Functions

- void [__gnu_pbds::__throw_container_error](#) (void)
- void [__gnu_pbds::__throw_insert_error](#) (void)
- void [__gnu_pbds::__throw_join_error](#) (void)
- void [__gnu_pbds::__throw_resize_error](#) (void)

6.133.1 Detailed Description

Contains exception classes.

Definition in file [exception.hpp](#).

6.134 exception_ptr.h File Reference

Classes

- class [std::__exception_ptr::exception_ptr](#)
An opaque pointer to an arbitrary exception.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Ex >
exception_ptr std::copy_exception (_Ex __ex) throw ()`
- `exception_ptr std::current_exception () throw ()`
- `template<typename _Ex >
exception_ptr std::make_exception_ptr (_Ex __ex) throw ()`
- `bool std::__exception_ptr::operator!= (const exception_ptr &, const
exception_ptr &) __attribute__((__pure__)) throw ()`
- `bool std::__exception_ptr::operator== (const exception_ptr &, const
exception_ptr &) __attribute__((__pure__)) throw ()`
- `void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::__exception_ptr::swap (exception_ptr &__lhs, exception_ptr &__-
rhs)`

6.134.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [exception_ptr.h](#).

6.135 extptr_allocator.h File Reference

Classes

- class [__gnu_cxx::_ExtPtr_allocator<_Tp>](#)
*An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Tp>
void __gnu_cxx::swap (_ExtPtr_allocator<_Tp> &__larg, _ExtPtr_allocator<_Tp> &__rarg)`

6.135.1 Detailed Description

Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

Definition in file [extptr_allocator.h](#).

6.136 features.h File Reference

Defines on whether to include algorithm variants.

Defines

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`

- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

6.136.1 Detailed Description

Defines on whether to include algorithm variants. Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [features.h](#).

6.136.2 Define Documentation

6.136.2.1 `#define _GLIBCXX_BAL_QUICKSORT`

Include parallel dynamically load-balanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file [features.h](#).

6.136.2.2 `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Include the equal-sized blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file [features.h](#).

6.136.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file [features.h](#).

6.136.2.4 #define _GLIBCXX_FIND_GROWING_BLOCKS

Include the growing blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file `features.h`.

6.136.2.5 #define _GLIBCXX_MERGESORT

Include parallel multi-way mergesort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

6.136.2.6 #define _GLIBCXX_QUICKSORT

Include parallel unbalanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file `features.h`.

6.136.2.7 #define _GLIBCXX_TREE_DYNAMIC_BALANCING

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file `features.h`.

6.136.2.8 `#define _GLIBCXX_TREE_FULL_COPY`

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file `features.h`.

6.136.2.9 `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

6.137 `fenv.h` File Reference

6.137.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fenv.h](#).

6.138 `find.h` File Reference

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`
`selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`
`selector, constant_size_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`
`selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`
`selector, equal_split_tag)`

6.138.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find.h](#).

6.139 find_selectors.h File Reference

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::__adjacent_find_selector](#)
Test predicate on two adjacent elements.
- struct [__gnu_parallel::__find_first_of_selector< _FIterator >](#)
Test predicate on several elements.
- struct [__gnu_parallel::__find_if_selector](#)
Test predicate on a single element, used for `std::find()` and `std::find_if()`.

- struct [__gnu_parallel::__generic_find_selector](#)
Base class of all [__gnu_parallel::__find_template](#) selectors.
- struct [__gnu_parallel::__mismatch_selector](#)
Test inverted predicate on a single element.

Namespaces

- namespace [__gnu_parallel](#)

6.139.1 Detailed Description

[_Function](#) objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find_selectors.h](#).

6.140 [for_each.h](#) File Reference

Main interface for embarrassingly parallel functions.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- template<typename [_Iter](#) , typename [_UserOp](#) , typename [_Functionality](#) , typename [_Red](#) , typename [_Result](#) >
[_UserOp](#) [__gnu_parallel::__for_each_template_random_access](#) ([_Iter](#) __begin, [_Iter](#) __end, [_UserOp](#) __user_op, [_Functionality](#) &__functionality, [_Red](#) __reduction, [_Result](#) __reduction_start, [_Result](#) &__output, typename [std::iterator_traits](#)< [_Iter](#) >::difference_type __bound, [_Parallelism](#) __parallelism_tag)

6.140.1 Detailed Description

Main interface for embarrassingly parallel functions. The explicit implementation are in other header files, like [workstealing.h](#), [par_loop.h](#), [omp_loop.h](#), and [omp_loop_static.h](#). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for_each.h](#).

6.141 for_each_selectors.h File Reference

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::__accumulate_binop_reduct< _BinOp >](#)
General reduction, using a binary operator.
- struct [__gnu_parallel::__accumulate_selector< _It >](#)
[std::accumulate\(\)](#) selector.
- struct [__gnu_parallel::__adjacent_difference_selector< _It >](#)
Selector that returns the difference between two adjacent `__elements`.
- struct [__gnu_parallel::__count_if_selector< _It, _Diff >](#)
[std::count_if\(\)](#) selector.
- struct [__gnu_parallel::__count_selector< _It, _Diff >](#)
[std::count\(\)](#) selector.
- struct [__gnu_parallel::__fill_selector< _It >](#)
[std::fill\(\)](#) selector.
- struct [__gnu_parallel::__for_each_selector< _It >](#)
[std::for_each\(\)](#) selector.
- struct [__gnu_parallel::__generate_selector< _It >](#)
[std::generate\(\)](#) selector.
- struct [__gnu_parallel::__generic_for_each_selector< _It >](#)
Generic `__selector` for embarrassingly parallel functions.
- struct [__gnu_parallel::__identity_selector< _It >](#)
Selector that just returns the passed iterator.
- struct [__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >](#)

[`std::inner_product\(\)`](#) selector.

- struct [`__gnu_parallel::__max_element_reduct< _Compare, _It >`](#)
Reduction for finding the maximum element, using a comparator.
- struct [`__gnu_parallel::__min_element_reduct< _Compare, _It >`](#)
Reduction for finding the maximum element, using a comparator.
- struct [`__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`](#)
`std::replace()` selector.
- struct [`__gnu_parallel::__replace_selector< _It, _Tp >`](#)
`std::replace()` selector.
- struct [`__gnu_parallel::__transform1_selector< _It >`](#)
`std::transform()` __selector, one input sequence variant.
- struct [`__gnu_parallel::__transform2_selector< _It >`](#)
`std::transform()` __selector, two input sequences variant.
- struct [`__gnu_parallel::__DummyReduct`](#)
Reduction function doing nothing.
- struct [`__gnu_parallel::__Nothing`](#)
Functor doing nothing.

Namespaces

- namespace [`__gnu_parallel`](#)

6.141.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for_each_selectors.h](#).

6.142 formatter.h File Reference

Classes

- struct [__gnu_debug::__is_same<_Type1, _Type2>](#)

Namespaces

- namespace [__gnu_debug](#)

Enumerations

- enum [_Debug_msg_id](#) {
[__msg_valid_range](#), [__msg_insert_singular](#), [__msg_insert_different](#), [__msg_erase_bad](#),
[__msg_erase_different](#), [__msg_subscript_oob](#), [__msg_empty](#), [__msg_unpartitioned](#),
[__msg_unpartitioned_pred](#), [__msg_unsorted](#), [__msg_unsorted_pred](#), [__msg_not_heap](#),
[__msg_not_heap_pred](#), [__msg_bad_bitset_write](#), [__msg_bad_bitset_read](#),
[__msg_bad_bitset_flip](#),
[__msg_self_splice](#), [__msg_splice_alloc](#), [__msg_splice_bad](#), [__msg_splice_other](#),
[__msg_splice_overlap](#), [__msg_init_singular](#), [__msg_init_copy_singular](#), [__msg_init_const_singular](#),
[__msg_copy_singular](#), [__msg_bad_deref](#), [__msg_bad_inc](#), [__msg_bad_dec](#),
[__msg_iter_subscript_oob](#), [__msg_advance_oob](#), [__msg_retreat_oob](#), [__msg_iter_compare_bad](#),
[__msg_compare_different](#), [__msg_iter_order_bad](#), [__msg_order_different](#),
[__msg_distance_bad](#),
[__msg_distance_different](#), [__msg_deref_istream](#), [__msg_inc_istream](#), [__msg_output_ostream](#),
[__msg_deref_istreambuf](#), [__msg_inc_istreambuf](#) }

6.142.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [formatter.h](#).

6.143 forward_list.h File Reference

Classes

- struct [std::_Fwd_list_base< _Tp, _Alloc >](#)
Base class for forward_list.
- struct [std::_Fwd_list_const_iterator< _Tp >](#)
A forward_list::const_iterator.
- struct [std::_Fwd_list_iterator< _Tp >](#)
A forward_list::iterator.
- struct [std::_Fwd_list_node< _Tp >](#)
A helper node class for forward_list. This is just a linked list with a data value in each node. There is a sorting utility method.
- struct [std::_Fwd_list_node_base](#)
A helper basic node class for forward_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.
- class [std::forward_list< _Tp, _Alloc >](#)
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Namespaces

- namespace [std](#)

Functions

- template<typename _Tp >
bool [std::operator!=](#) (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_iterator< _Tp > &__y)
- template<typename _Tp, typename _Alloc >
bool [std::operator!=](#) (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc >
bool [std::operator<](#) (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp >`
`bool std::operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`

6.143.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward_list.h](#).

6.144 forward_list.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

6.144.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward_list.tcc](#).

6.145 fstream File Reference

Classes

- class [std::basic_filebuf< _CharT, _Traits >](#)

*The actual work of input and output (for files).
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.*

- class [std::basic_fstream< _CharT, _Traits >](#)

*Controlling input and output for files.
This class supports reading from and writing to named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.*

- class [std::basic_ifstream< _CharT, _Traits >](#)

*Controlling input for files.
This class supports reading from named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.*

- class [std::basic_ofstream< _CharT, _Traits >](#)

*Controlling output for files.
This class supports reading from named files, using the inherited functions from [std::basic_ostream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.*

Namespaces

- namespace [std](#)

6.145.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fstream](#).

6.146 fstream.tcc File Reference

Namespaces

- namespace [std](#)

6.146.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [fstream.tcc](#).

6.147 funtexcept.h File Reference

Namespaces

- namespace [std](#)

Functions

- void **std::__throw_bad_alloc** (void) __attribute__((__noreturn__))
- void **std::__throw_bad_cast** (void) __attribute__((__noreturn__))
- void **std::__throw_bad_exception** (void) __attribute__((__noreturn__))
- void **std::__throw_bad_function_call** () __attribute__((__noreturn__))
- void **std::__throw_bad_typeid** (void) __attribute__((__noreturn__))
- void **std::__throw_domain_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_future_error** (int) __attribute__((__noreturn__))
- void **std::__throw_invalid_argument** (const char *) __attribute__((__noreturn__))
- void **std::__throw_ios_failure** (const char *) __attribute__((__noreturn__))
- void **std::__throw_length_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_logic_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_out_of_range** (const char *) __attribute__((__noreturn__))
- void **std::__throw_overflow_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_range_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_runtime_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_system_error** (int) __attribute__((__noreturn__))
- void **std::__throw_underflow_error** (const char *) __attribute__((__noreturn__))

6.147.1 Detailed Description

This header provides support for `-fno-exceptions`.

Definition in file [functexcept.h](#).

6.148 functional File Reference

Classes

- struct [std::__is_location_invariant< _Tp >](#)
- struct [std::_Derives_from_binary_function< _Tp >](#)
Determines if the type `_Tp` derives from `binary_function`.
- struct [std::_Derives_from_unary_function< _Tp >](#)
Determines if the type `_Tp` derives from `unary_function`.
- class [std::_Function_base](#)
Base class of all polymorphic function object wrappers.
- struct [std::_Function_to_function_pointer< _Tp, _IsFunctionType >](#)
Turns a function type into a function pointer type.
- class [std::_Has_result_type_helper< _Tp >](#)
- struct [std::_Maybe_get_result_type< _Has_result_type, _Functor >](#)
If we have found a `result_type`, extract it.
- struct [std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >](#)
- struct [std::_Maybe_unary_or_binary_function< _Res, _T1 >](#)
Derives from `unary_function`, as appropriate.
- struct [std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >](#)
Derives from `binary_function`, as appropriate.
- struct [std::_Maybe_wrap_member_pointer< _Tp >](#)
- struct [std::_Maybe_wrap_member_pointer< _Tp _Class::* >](#)
- class [std::_Mem_fn< _Res\(_Class::*\)\(_ArgTypes...\) const >](#)
Implementation of `mem_fn` for const member function pointers.
- class [std::_Mem_fn< _Res\(_Class::*\)\(_ArgTypes...\) const volatile >](#)
Implementation of `mem_fn` for const volatile member function pointers.

- class `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >`
Implementation of `mem_fn` for volatile member function pointers.
- class `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) >`
Implementation of `mem_fn` for member function pointers.
- class `std::_Mu< _Arg, false, false >`
- class `std::_Mu< _Arg, false, true >`
- class `std::_Mu< _Arg, true, false >`
- class `std::_Mu< reference_wrapper< _Tp >, false, false >`
- struct `std::_Placeholder< _Num >`
The type of placeholder objects defined by `libstdc++`.
- struct `std::_Reference_wrapper_base< _Tp >`
- struct `std::_Safe_tuple_element< __i, _Tuple >`
- struct `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`
- struct `std::_Safe_tuple_element_impl< __i, _Tuple, false >`
- struct `std::_Weak_result_type< _Functor >`
- struct `std::_Weak_result_type_impl< _Functor >`
- struct `std::_Weak_result_type_impl< _Res(&)(_ArgTypes...) >`
Retrieve the result type for a function reference.
- struct `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...) >`
Retrieve the result type for a function pointer.
- struct `std::_Weak_result_type_impl< _Res(_ArgTypes...) >`
Retrieve the result type for a function type.
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >`
Retrieve result type for a const member function pointer.
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >`
Retrieve result type for a const volatile member function pointer.
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >`
Retrieve result type for a volatile member function pointer.
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) >`
Retrieve result type for a member function pointer.
- class `std::bad_function_call`

Exception class thrown when class template function's operator() is called with an empty target.

- class [std::function< _Res\(_ArgTypes...\)>](#)
*Primary class template for std::function.
 Polymorphic function wrapper.*
- struct [std::is_bind_expression< _Tp >](#)
Determines if the given type _Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].
- struct [std::is_bind_expression< _Bind< _Signature > >](#)
Class template _Bind is always a bind expression.
- struct [std::is_bind_expression< _Bind_result< _Result, _Signature > >](#)
Class template _Bind is always a bind expression.
- struct [std::is_placeholder< _Tp >](#)
Determines if the given type _Tp is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct [std::is_placeholder< _Placeholder< _Num > >](#)
- class [std::reference_wrapper< _Tp >](#)
Primary class template for [reference_wrapper](#).

Namespaces

- namespace [std](#)
- namespace [std::placeholders](#)

Enumerations

- enum [_Manager_operation](#) { [__get_type_info](#), [__get_functor_ptr](#), [__clone_functor](#), [__destroy_functor](#) }

Functions

- template<typename _Functor >
[_Functor & std::__callable_functor](#) (_Functor &__f)
- template<typename _Member , typename _Class >
[_Mem_fn< _Member _Class::* > std::__callable_functor](#) (_Member _Class::*&__p)

- `template<typename _Member, typename _Class >
_Mem_fn< _Member _Class::* > std::__callable_function (_Member _-
Class::*const &__p)`
- `template<typename _Functor, typename... _ArgTypes>
_Bind< typename _Maybe_wrap_member_pointer< _Functor >::type(_-
ArgTypes...)> std::bind (_Functor __f, _ArgTypes...__args)`
- `template<typename _Result, typename _Functor, typename... _ArgTypes>
_Bind_result< _Result, typename _Maybe_wrap_member_pointer< _Functor
>::type(_ArgTypes...)> std::bind (_Functor __f, _ArgTypes...__args)`
- `template<typename _Tp, typename _Class >
_Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm)`
- `template<typename _Res, typename... _Args>
bool std::operator!= (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>
bool std::operator!= (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>
bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>
bool std::operator== (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>
void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)>
&__y)`
- `template<typename _Tp >
reference_wrapper< _Tp > std::ref (_Tp &__t)`
- `template<typename _Tp >
reference_wrapper< const _Tp > std::cref (const _Tp &__t)`
- `template<typename _Tp >
reference_wrapper< _Tp > std::ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >
reference_wrapper< const _Tp > std::cref (reference_wrapper< _Tp > __t)`

Variables

- `enable_if< (!is_member_pointer< _Functor >::value &&!is_function< _-
Functor >::value &&!is_function< typename remove_pointer< _Functor
>::type >::value), typename result_of< _Functor(_Args...)>::type >::type
std::__invoke (_Functor &__f, _Args &&...__args)`

6.148.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [functional](#).

6.149 functional File Reference

Classes

- class [__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >](#)
An SGI extension .
- struct [__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >](#)
An SGI extension .
- struct [__gnu_cxx::constant_unary_fun< _Result, _Argument >](#)
An SGI extension .
- struct [__gnu_cxx::constant_void_fun< _Result >](#)
An SGI extension .
- struct [__gnu_cxx::project1st< _Arg1, _Arg2 >](#)
An SGI extension .
- struct [__gnu_cxx::project2nd< _Arg1, _Arg2 >](#)
An SGI extension .
- struct [__gnu_cxx::select1st< _Pair >](#)
An SGI extension .
- struct [__gnu_cxx::select2nd< _Pair >](#)
An SGI extension .
- class [__gnu_cxx::subtractive_rng](#)
- class [__gnu_cxx::unary_compose< _Operation1, _Operation2 >](#)
An SGI extension .

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<class _Operation1 , class _Operation2 >`
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _-`
`Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_-`
`cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const`
`_Operation3 &__fn3)`
- `template<class _Result >`
`constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result`
`&__val)`
- `template<class _Result >`
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const`
`_Result &__val)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<class _Ret , class _Tp , class _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > __gnu_cxx::mem_fun1 (_Ret(_Tp::*__f)(_-`
`Arg))`
- `template<class _Ret , class _Tp , class _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > __gnu_cxx::mem_fun1_ref (_Ret(_-`
`Tp::*__f)(_Arg))`

6.149.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/functional](#).

6.150 functional_hash.h File Reference

Classes

- `struct std::hash< _Tp >`
Primary class template hash.
- `struct std::hash< _Tp * >`

Partial specializations for pointer types.

Namespaces

- namespace [std](#)

Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

6.150.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [functional_hash.h](#).

6.151 functions.h File Reference

Namespaces

- namespace [__gnu_debug](#)

Functions

- `template<typename _Iterator >`
`bool __gnu_debug::__check_dereferenceable (_Iterator &)`
- `template<typename _Tp >`
`bool __gnu_debug::__check_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`
`bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`
`bool __gnu_debug::__check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`

- `template<typename _ForwardIterator, typename _Tp >`
`bool __gnu_debug::__check_partitioned_upper (_ForwardIterator __first, _-`
`ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iterator >`
`bool __gnu_debug::__check_singular (_Iterator &__x)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__check_singular (const _Safe_iterator< _Iterator, _-`
`Sequence > &__x)`
- `template<typename _Tp >`
`bool __gnu_debug::__check_singular (const _Tp *__ptr)`
- `bool __gnu_debug::__check_singular_aux (const void *)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _-`
`InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _-`
`InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _-`
`InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`
`ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _-`
`InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`
`ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`
`_InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`
`_InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`
`const _InputIterator &__last, _Predicate __pred, std::__true_type)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`
`const _InputIterator &__last, std::__true_type)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`
`InputIterator &, std::__false_type)`

- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`
`InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT >`
`const _CharT * __gnu_debug::__check_string (const _CharT *__s)`
- `template<typename _CharT, typename _Integer >`
`const _CharT * __gnu_debug::__check_string (const _CharT *__s, const _-`
`Integer &__n __attribute__((__unused__)))`
- `template<typename _InputIterator >`
`_InputIterator __gnu_debug::__check_valid_range (const _InputIterator &__-`
`first, const _InputIterator &__last __attribute__((__unused__)))`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence`
`> &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__valid_range (const _InputIterator &__first, const _-`
`InputIterator &__last)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__valid_range_aux (const _InputIterator &__first, const _-`
`InputIterator &__last, std::__false_type)`
- `template<typename _Integral >`
`bool __gnu_debug::__valid_range_aux (const _Integral &, const _Integral &,`
`std::__true_type)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__valid_range_aux2 (const _InputIterator &, const _-`
`InputIterator &, std::input_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator &__-`
`first, const _RandomAccessIterator &__last, std::random_access_iterator_tag)`

6.151.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [functions.h](#).

6.152 future File Reference

Classes

- class `std::__basic_future< _Res >`
Common implementation for future and [shared_future](#).

- struct `std::__future_base`
Base class and enclosing scope.
- struct `std::__future_base::Ptr<_Res>`
A `unique_ptr` based on the instantiating type.
- struct `std::__future_base::Result<_Res>`
Result.
- struct `std::__future_base::Result<_Res &>`
Partial specialization for reference types.
- struct `std::__future_base::Result<void>`
Explicit specialization for void.
- struct `std::__future_base::Result_base`
Base class for results.
- class `std::__future_base::State`
Shared state between a promise and one or more associated futures.
- class `std::future<_Res>`
Primary template for future.
- class `std::future<_Res &>`
Partial specialization for `future<R&>`
- class `std::future<void>`
Explicit specialization for `future<void>`
- class `std::future_error`
Exception type thrown by futures.
- class `std::packaged_task<_Res(_ArgTypes...)>`
`packaged_task`
- class `std::promise<_Res>`
Primary template for promise.
- class `std::promise<_Res &>`
Partial specialization for `promise<R&>`

- class `std::promise< void >`
Explicit specialization for `promise<void>`
- class `std::shared_future< _Res >`
Primary template for `shared_future`.
- class `std::shared_future< _Res & >`
Partial specialization for `shared_future<R&>`
- class `std::shared_future< void >`
Explicit specialization for `shared_future<void>`

Namespaces

- namespace `std`

Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }

Functions

- template<typename _Fn, typename... _Args>
future< typename result_of< _Fn(_Args...)>::type > **std::async** (launch __-
policy, _Fn &&__fn, _Args &&...__args)
- template<typename _Fn, typename... _Args>
enable_if<!is_same< typename decay< _Fn >::type, launch >::value, future<
decltype(std::declval< _Fn >)(std::declval< _Args >)...)> >::type **std::async**
(_Fn &&__fn, _Args &&...__args)
- error_code **std::make_error_code** (future_errc __errc)
- error_condition **std::make_error_condition** (future_errc __errc)
- template<typename _Res >
void **std::swap** (promise< _Res > &__x, promise< _Res > &__y)
- template<typename _Res, typename... _ArgTypes>
void **std::swap** (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task<
_Res(_ArgTypes...)> &__y)

Variables

- `const error_category *const std::future_category`

6.152.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [future](#).

6.153 `gslice.h` File Reference

Classes

- class [std::gslice](#)
Class defining multi-dimensional subset of an array.

Namespaces

- namespace [std](#)

6.153.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice.h](#).

6.154 `gslice_array.h` File Reference

Classes

- class [std::gslice_array<_Tp>](#)
Reference to multi-dimensional subset of an array.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

6.154.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice_array.h](#).

6.155 hash_fun.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `size_t __gnu_cxx::__stl_hash_string (const char *__s)`

6.155.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash_fun.h](#).

6.156 hash_map File Reference

Classes

- class [__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >](#)
- class [__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool __gnu_cxx::operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool __gnu_cxx::operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool __gnu_cxx::operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool __gnu_cxx::operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`void __gnu_cxx::swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`void __gnu_cxx::swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`

6.156.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash_map](#).

6.157 hash_policy.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

6.157.1 Detailed Description

Contains hash-related policies.

Definition in file [hash_policy.hpp](#).

6.158 `hash_set` File Reference

Classes

- class [__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc >](#)
- class [__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool __gnu_cxx::operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool __gnu_cxx::operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool __gnu_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool __gnu_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`void __gnu_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`void __gnu_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

6.158.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash_set](#).

6.159 hashtable.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Enumerations

- enum { [_S_num_primes](#) }

Functions

- unsigned long **__gnu_cxx::__stl_next_prime** (unsigned long __n)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >
bool **__gnu_cxx::operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >
bool **__gnu_cxx::operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >
void **__gnu_cxx::swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)

Variables

- static const unsigned long **__gnu_cxx::__stl_prime_list** [_S_num_primes]

6.159.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [backward/hashtable.h](#).

6.160 hashtable.h File Reference

Namespaces

- namespace [std](#)

6.160.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/hashtable.h](#).

6.161 hashtable_policy.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Functions

- `template<class _Iterator >
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw
(_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw
(_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw
(_Iterator __first, _Iterator __last)`
- `template<typename _Value , bool __cache>
bool std::__detail::operator!= (const _Node_iterator_base< _Value, __cache
> &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>
bool std::__detail::operator!= (const _Hashtable_iterator_base< _Value, __-
cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>
bool std::__detail::operator== (const _Hashtable_iterator_base< _Value, __-
cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>
bool std::__detail::operator== (const _Node_iterator_base< _Value, __cache
> &__x, const _Node_iterator_base< _Value, __cache > &__y)`

Variables

- `const unsigned long std::__detail::__prime_list []`

6.161.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [hashtable_policy.h](#).

6.162 `indirect_array.h` File Reference

Classes

- class [std::indirect_array<_Tp>](#)
Reference to arbitrary subset of an array.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

6.162.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [indirect_array.h](#).

6.163 `initializer_list` File Reference

Classes

- class [std::initializer_list<_E>](#)
initializer_list

Namespaces

- namespace [std](#)

6.163.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [initializer_list](#).

6.164 iomanip File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _MoneyT >`
`_Get_money< _MoneyT > std::get_money (_MoneyT &__mon, bool __-`
`intl=false)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__is, _Setprecision __f)`
- `template<typename _MoneyT >`
`_Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __-`
`intl=false)`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`
- `template<typename _CharT >`
`_Setfill< _CharT > std::setfill (_CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

6.164.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iomanip](#).

6.165 ios File Reference

6.165.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ios](#).

6.166 ios_base.h File Reference

Classes

- class [std::ios_base](#)

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see [ios_base](#) when they need to specify the full name of the various I/O flags (e.g., the openmodes).

- class [std::ios_base::failure](#)

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios_base::failure](#).

Namespaces

- namespace [std](#)

Defines

- `#define _IOS_BASE_SEEK_CUR`
- `#define _IOS_BASE_SEEK_END`

Enumerations

- enum `_Ios_Fmtflags` {
`_S_boolalpha, _S_dec, _S_fixed, _S_hex,`
`_S_internal, _S_left, _S_oct, _S_right,`
`_S_scientific, _S_showbase, _S_showpoint, _S_showpos,`
`_S_skipws, _S_unitbuf, _S_uppercase, _S_adjustfield,`
`_S_basefield, _S_floatfield, _S_ios_fmtflags_end` }
- enum `_Ios_Iostate` {
`_S_goodbit, _S_badbit, _S_eofbit, _S_failbit,`
`_S_ios_iostate_end` }
- enum `_Ios_Openmode` {
`_S_app, _S_ate, _S_bin, _S_in,`
`_S_out, _S_trunc, _S_ios_openmode_end` }
- enum `_Ios_Seekdir` { `_S_beg, _S_cur, _S_end, _S_ios_seekdir_end` }

Functions

- `ios_base & std::boolalpha (ios_base &__base)`
- `ios_base & std::dec (ios_base &__base)`
- `ios_base & std::fixed (ios_base &__base)`

- `ios_base & std::hex (ios_base &__base)`
- `ios_base & std::internal (ios_base &__base)`
- `ios_base & std::left (ios_base &__base)`
- `ios_base & std::noboolalpha (ios_base &__base)`
- `ios_base & std::noshowbase (ios_base &__base)`
- `ios_base & std::noshowpoint (ios_base &__base)`
- `ios_base & std::noshowpos (ios_base &__base)`
- `ios_base & std::noskipws (ios_base &__base)`
- `ios_base & std::nounitbuf (ios_base &__base)`
- `ios_base & std::nouppercase (ios_base &__base)`
- `ios_base & std::oct (ios_base &__base)`
- `_Ios_Fmtflags std::operator& (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Openmode std::operator& (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Iostate std::operator& (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & std::operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Iostate & std::operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Openmode & std::operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate std::operator^ (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Openmode std::operator^ (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags std::operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Iostate & std::operator^= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & std::operator^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Openmode & std::operator^= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate std::operator| (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Fmtflags std::operator| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Openmode std::operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags & std::operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Openmode & std::operator|= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & std::operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Iostate std::operator~ (_Ios_Iostate __a)`
- `_Ios_Fmtflags std::operator~ (_Ios_Fmtflags __a)`
- `_Ios_Openmode std::operator~ (_Ios_Openmode __a)`
- `ios_base & std::right (ios_base &__base)`
- `ios_base & std::scientific (ios_base &__base)`
- `ios_base & std::showbase (ios_base &__base)`
- `ios_base & std::showpoint (ios_base &__base)`
- `ios_base & std::showpos (ios_base &__base)`
- `ios_base & std::skipws (ios_base &__base)`
- `ios_base & std::unitbuf (ios_base &__base)`
- `ios_base & std::uppercase (ios_base &__base)`

6.166.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ios_base.h](#).

6.167 iosfwd File Reference

Namespaces

- namespace [std](#)

Typedefs

- typedef basic_filebuf< char > [std::filebuf](#)
- typedef basic_fstream< char > [std::fstream](#)
- typedef basic_ifstream< char > [std::ifstream](#)
- typedef basic_ios< char > [std::ios](#)
- typedef basic_iostream< char > [std::iostream](#)
- typedef basic_istream< char > [std::istream](#)
- typedef basic_istreamstream< char > [std::istreamstream](#)
- typedef basic_ofstream< char > [std::ofstream](#)
- typedef basic_ostream< char > [std::ostream](#)
- typedef basic_ostringstream< char > [std::ostringstream](#)
- typedef basic_streambuf< char > [std::streambuf](#)
- typedef basic_stringbuf< char > [std::stringbuf](#)
- typedef basic_stringstream< char > [std::stringstream](#)
- typedef basic_filebuf< wchar_t > [std::wfilebuf](#)
- typedef basic_fstream< wchar_t > [std::wfstream](#)
- typedef basic_ifstream< wchar_t > [std::wifstream](#)
- typedef basic_ios< wchar_t > [std::wios](#)
- typedef basic_iostream< wchar_t > [std::wiostream](#)
- typedef basic_istream< wchar_t > [std::wistream](#)
- typedef basic_istreamstream< wchar_t > [std::wistreamstream](#)
- typedef basic_ofstream< wchar_t > [std::wofstream](#)
- typedef basic_ostream< wchar_t > [std::wostream](#)
- typedef basic_ostringstream< wchar_t > [std::wostringstream](#)
- typedef basic_streambuf< wchar_t > [std::wstreambuf](#)
- typedef basic_stringbuf< wchar_t > [std::wstringbuf](#)
- typedef basic_stringstream< wchar_t > [std::wstringstream](#)

6.167.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iosfwd](#).

6.168 iostream File Reference

Namespaces

- namespace [std](#)

Variables

- static `ios_base::Init` **std::__ioinit**

Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's FILE streams, and to be available during program startup and termination. For more information, see the HOWTO linked to above.

- `istream` [std::cin](#)
- `ostream` [std::cout](#)
- `ostream` [std::cerr](#)
- `ostream` [std::clog](#)
- `wistream` [std::wcin](#)
- `wostream` [std::wcout](#)
- `wostream` [std::wcerr](#)
- `wostream` [std::wclog](#)

6.168.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iostream](#).

6.169 istream File Reference

Classes

- class `std::basic_istream< _CharT, _Traits >`
Merging istream and ostream capabilities.
This class multiply inherits from the input and output stream classes simply to provide a single interface.
- class `std::basic_istream< _CharT, _Traits >`
Controlling input.
This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.
- class `std::basic_istream< _CharT, _Traits >::sentry`
Performs setup work for input streams.

Namespaces

- namespace `std`

Functions

- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &&__is, _Tp &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__in, _CharT *__s)`
- `template<>`
`basic_istream< char > & std::operator>> (basic_istream< char > &__in,`
`char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _-`
`Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _-`
`Traits > &__in, signed char *__s)`

6.169.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [istream](#).

6.170 istream.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits`
`> &__is)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__in, _CharT &__c)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`
`CharT, _Traits > &__in, _CharT *__s)`

6.170.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [istream.tcc](#).

6.171 iterator File Reference

6.171.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iterator](#).

6.172 iterator File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _InputIterator, typename _Distance >
void __gnu_cxx::__distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >
void __gnu_cxx::__distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Distance >
void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

6.172.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/iterator](#).

6.173 iterator.h File Reference

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- class [__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >](#)
A pair of iterators. The usual iterator operations are applied to both child iterators.
- class [__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >](#)
A triple of iterators. The usual iterator operations are applied to all three child iterators.

Namespaces

- namespace [__gnu_parallel](#)

6.173.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [iterator.h](#).

6.174 iterator_tracker.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >
bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (type-`
`name __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const`
`__iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence >::difference_type std::__-`
`profile::operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__-`
`profile::operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _-`
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`
`rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence`
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _-`
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _-`
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`
`rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool std::__profile::operator== (const __iterator_tracker< _Iterator, _-`
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _-`
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`
`rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _-`
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`
`rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence`
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _-`
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _-`
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`
`rhs)`

6.174.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [iterator_tracker.h](#).

6.175 limits File Reference

Classes

- struct [std::__numeric_limits_base](#)
Part of [std::numeric_limits](#).
- struct [std::numeric_limits< _Tp >](#)
Properties of fundamental types.
- struct [std::numeric_limits< bool >](#)
[numeric_limits<bool>](#) specialization.
- struct [std::numeric_limits< char >](#)
[numeric_limits<char>](#) specialization.
- struct [std::numeric_limits< char16_t >](#)
[numeric_limits<char16_t>](#) specialization.
- struct [std::numeric_limits< char32_t >](#)
[numeric_limits<char32_t>](#) specialization.
- struct [std::numeric_limits< double >](#)
[numeric_limits<double>](#) specialization.
- struct [std::numeric_limits< float >](#)
[numeric_limits<float>](#) specialization.
- struct [std::numeric_limits< int >](#)
[numeric_limits<int>](#) specialization.

- struct `std::numeric_limits< long >`
numeric_limits<long> specialization.
- struct `std::numeric_limits< long double >`
numeric_limits<long double> specialization.
- struct `std::numeric_limits< long long >`
numeric_limits<long long> specialization.
- struct `std::numeric_limits< short >`
numeric_limits<short> specialization.
- struct `std::numeric_limits< signed char >`
numeric_limits<signed char> specialization.
- struct `std::numeric_limits< unsigned char >`
numeric_limits<unsigned char> specialization.
- struct `std::numeric_limits< unsigned int >`
numeric_limits<unsigned int> specialization.
- struct `std::numeric_limits< unsigned long >`
numeric_limits<unsigned long> specialization.
- struct `std::numeric_limits< unsigned long long >`
numeric_limits<unsigned long long> specialization.
- struct `std::numeric_limits< unsigned short >`
numeric_limits<unsigned short> specialization.
- struct `std::numeric_limits< wchar_t >`
numeric_limits<wchar_t> specialization.

Namespaces

- namespace `std`

Defines

- `#define __glibcxx_digits(T)`
- `#define __glibcxx_digits10(T)`
- `#define __glibcxx_double_has_denorm_loss`
- `#define __glibcxx_double_tinyness_before`
- `#define __glibcxx_double_traps`
- `#define __glibcxx_float_has_denorm_loss`
- `#define __glibcxx_float_tinyness_before`
- `#define __glibcxx_float_traps`
- `#define __glibcxx_integral_traps`
- `#define __glibcxx_long_double_has_denorm_loss`
- `#define __glibcxx_long_double_tinyness_before`
- `#define __glibcxx_long_double_traps`
- `#define __glibcxx_max(T)`
- `#define __glibcxx_max_digits10(T)`
- `#define __glibcxx_min(T)`
- `#define __glibcxx_signed(T)`

Enumerations

- `enum std::float_denorm_style { std::denorm_indeterminate, std::denorm_absent, std::denorm_present }`
- `enum std::float_round_style {
std::round_indeterminate, std::round_toward_zero, std::round_to_nearest,
std::round_toward_infinity,
std::round_toward_neg_infinity }`

6.175.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [limits](#).

6.176 list File Reference

6.176.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [list](#).

6.177 list File Reference

Classes

- class [std::__debug::list< _Tp, _Allocator >](#)
Class [std::list](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Tp, typename _Alloc >
bool std::__debug::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

6.177.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/list](#).

6.178 list File Reference

Classes

- class [std::__profile::list< _Tp, _Allocator >](#)
List wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

6.178.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/list](#).

6.179 list.tcc File Reference

Namespaces

- namespace [std](#)

6.179.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [list.tcc](#).

6.180 list_partition.h File Reference

__Functionality to split __sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter >
void __gnu_parallel::__shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >
void __gnu_parallel::__shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >
size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`

6.180.1 Detailed Description

__Functionality to split __sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [list_partition.h](#).

6.181 list_update_policy.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

6.181.1 Detailed Description

Contains policies for list update containers.

Definition in file [list_update_policy.hpp](#).

6.182 locale File Reference

6.182.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [locale](#).

6.183 locale_classes.h File Reference

Classes

- class [std::collate<_CharT>](#)
Facet for localized string comparison.
- class [std::collate_byname<_CharT>](#)
class [collate_byname](#) [22.2.4.2].
- class [std::locale](#)

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

- class [std::locale::facet](#)

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class [std::locale::id](#)

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member; otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Facet >
bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _Facet >
const _Facet & std::use_facet (const locale &__loc)`

6.183.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale_classes.h](#).

6.184 locale_classes.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Facet >`
`bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`
`const _Facet & std::use_facet (const locale &__loc)`

6.184.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale_classes.tcc](#).

6.185 [locale_facets.h](#) File Reference

Classes

- class [std::__ctype_abstract_base< _CharT >](#)
Common base for ctype facet.
- class [std::ctype< _CharT >](#)
Primary class template ctype facet.
This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.
- class [std::ctype< char >](#)
The [ctype<char>](#) specialization.
This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.
- class [std::ctype< wchar_t >](#)
The [ctype<wchar_t>](#) specialization.
This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
- class [std::ctype_byname< _CharT >](#)
class [ctype_byname](#) [22.2.1.2].
- class [std::ctype_byname< char >](#)
22.2.1.4 Class [ctype_byname](#) specializations.

- class `std::num_get<_CharT, _InIter >`
*Primary class template `num_get`.
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class `std::num_put<_CharT, _OutIter >`
*Primary class template `num_put`.
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- class `std::numpunct<_CharT >`
*Primary class template `numpunct`.
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class `std::numpunct_byname<_CharT >`
class `numpunct_byname` [22.2.3.2].

Namespaces

- namespace `std`

Defines

- `#define _GLIBCXX_NUM_FACETS`

Functions

- `template<typename _CharT >`
`_CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __-
gbeg, size_t __gsize, const _CharT * __first, const _CharT * __last)`
- `template<typename _Tp >`
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __-
c_locale &) throw ()`
- `template<>`
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __-
c_locale &) throw ()`

- `template<>`
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT >`
`ostreambuf_iterator< _CharT > std::__write (ostreambuf_iterator< _CharT > __s, const _CharT * __ws, int __len)`
- `template<typename _CharT, typename _OutIter >`
`_OutIter std::__write (_OutIter __s, const _CharT * __ws, int __len)`
- `template<typename _CharT >`
`bool std::isalnum (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isalpha (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isctrl (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isgraph (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::islower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isprint (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::ispunct (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isspace (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isupper (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isxdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`_CharT std::tolower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`_CharT std::toupper (_CharT __c, const locale & __loc)`

6.185.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale_facets.h](#).

6.186 locale_facets.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT >
_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__-
gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _CharT, typename _ValueT >
_GLIBCXX_END_LDBL_NAMESPACE int std::__int_to_char (_CharT *__-
bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __-
dec)`
- `_GLIBCXX_PURE bool std::__verify_grouping (const char *__grouping,
size_t __grouping_size, const string &__grouping_tmp) throw ()`

6.186.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale_facets.tcc](#).

6.187 locale_facets_nonio.h File Reference

Classes

- class [std::messages<_CharT>](#)
*Primary class template messages.
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [std::messages_base](#)
Messages facet base class providing catalog typedef.
- class [std::messages_byname<_CharT>](#)
class messages_byname [22.2.7.2].
- class [std::money_base](#)

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

- class `std::money_get< _CharT, _InIter >`

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

- class `std::money_put< _CharT, _OutIter >`

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount.

- class `std::moneypunct< _CharT, _Intl >`

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class `std::moneypunct_byname< _CharT, _Intl >`

class `moneypunct_byname` [22.2.6.4].

- class `std::time_base`

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

- class `std::time_get< _CharT, _InIter >`

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class `std::time_get_byname< _CharT, _InIter >`

class `time_get_byname` [22.2.5.2].

- class `std::time_put< _CharT, _OutIter >`

Primary class template `time_put`.

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

- class `std::time_put_byname< _CharT, _OutIter >`

class `time_put_byname` [22.2.5.4].

Namespaces

- namespace [std](#)

6.187.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale_facets_nonio.h](#).

6.188 locale_facets_nonio.tcc File Reference

Namespaces

- namespace [std](#)

6.188.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale_facets_nonio.tcc](#).

6.189 localefwd.h File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Facet >
bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _CharT >
bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >
bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >
bool std::isctrl (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet >`
`const _Facet & std::use_facet (const locale &__loc)`

6.189.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [localefwd.h](#).

6.190 losertree.h File Reference

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`
Stable [_LoserTree](#) variant.
- class `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`
Unstable [_LoserTree](#) variant.

- class `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`
Guarded loser/tournament tree.
- struct `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`
Internal representation of a `_LoserTree` element.
- class `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`
Stable `_LoserTree` implementation.
- class `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`
Unstable `_LoserTree` implementation.
- class `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`
Base class of `_LoserTree` implementation using pointers.
- struct `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`
Internal representation of `_LoserTree` __elements.
- class `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`
Stable unguarded `_LoserTree` variant storing pointers.
- class `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`
Unstable unguarded `_LoserTree` variant storing pointers.
- class `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`
Unguarded loser tree, keeping only pointers to the elements in the tree structure.
- class `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`
Stable implementation of unguarded `_LoserTree`.
- class `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`
Non-Stable implementation of unguarded `_LoserTree`.
- class `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`
Base class for unguarded `_LoserTree` implementation.

Namespaces

- namespace `__gnu_parallel`

6.190.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [losertree.h](#).

6.191 macros.h File Reference

Defines

- #define [__glibcxx_check_erase](#)(_Position)
- #define [__glibcxx_check_erase_range](#)(_First, _Last)
- #define [__glibcxx_check_heap](#)(_First, _Last)
- #define [__glibcxx_check_heap_pred](#)(_First, _Last, _Pred)
- #define [__glibcxx_check_insert](#)(_Position)
- #define [__glibcxx_check_insert_range](#)(_Position, _First, _Last)
- #define [__glibcxx_check_nonempty](#)()
- #define [__glibcxx_check_partitioned_lower](#)(_First, _Last, _Value)
- #define [__glibcxx_check_partitioned_lower_pred](#)(_First, _Last, _Value, _Pred)
- #define [__glibcxx_check_partitioned_upper](#)(_First, _Last, _Value)
- #define [__glibcxx_check_partitioned_upper_pred](#)(_First, _Last, _Value, _Pred)
- #define [__glibcxx_check_sorted](#)(_First, _Last)
- #define [__glibcxx_check_sorted_pred](#)(_First, _Last, _Pred)
- #define [__glibcxx_check_sorted_set](#)(_First1, _Last1, _First2)
- #define [__glibcxx_check_sorted_set_pred](#)(_First1, _Last1, _First2, _Pred)
- #define [__glibcxx_check_string](#)(_String)
- #define [__glibcxx_check_string_len](#)(_String, _Len)
- #define [__glibcxx_check_subscript](#)(_N)
- #define [__glibcxx_check_valid_range](#)(_First, _Last)
- #define [_GLIBCXX_DEBUG_VERIFY](#)(_Condition, _ErrorMessage)

6.191.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [macros.h](#).

6.191.2 Define Documentation

6.191.2.1 `#define __glibcxx_check_erase(_Position)`

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 102 of file macros.h.

6.191.2.2 `#define __glibcxx_check_erase_range(_First, _Last)`

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 116 of file macros.h.

6.191.2.3 `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.

Definition at line 229 of file macros.h.

6.191.2.4 `#define __glibcxx_check_insert(_Position)`

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end) and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 64 of file macros.h.

6.191.2.5 `#define __glibcxx_check_insert_range(_Position, _First, _Last)`

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Todo

We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Definition at line 87 of file macros.h.

6.191.2.6 **#define __glibcxx_check_partitioned_lower(*_First*, *_Last*, *_Value*)**

Verify that the iterator range [*_First*, *_Last*) is partitioned w.r.t. the value *_Value*.

Definition at line 177 of file macros.h.

6.191.2.7 **#define __glibcxx_check_partitioned_lower_pred(*_First*, *_Last*, *_Value*, *_Pred*)**

Verify that the iterator range [*_First*, *_Last*) is partitioned w.r.t. the value *_Value* and predicate *_Pred*.

Definition at line 197 of file macros.h.

6.191.2.8 **#define __glibcxx_check_partitioned_upper_pred(*_First*, *_Last*, *_Value*, *_Pred*)**

Verify that the iterator range [*_First*, *_Last*) is partitioned w.r.t. the value *_Value* and predicate *_Pred*.

Definition at line 209 of file macros.h.

6.191.2.9 **#define __glibcxx_check_sorted_pred(*_First*, *_Last*, *_Pred*)**

Verify that the iterator range [*_First*, *_Last*) is sorted by the predicate *_Pred*.

Definition at line 148 of file macros.h.

6.191.2.10 **#define _GLIBCXX_DEBUG_VERIFY(*_Condition*, *_ErrorMessage*)**

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 42 of file macros.h.

Referenced by `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::Safe_iterator()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator*()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator++()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator--()`, `__gnu_debug::Safe_iterator< _`

Iterator, _Sequence >::operator->(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

6.192 malloc_allocator.h File Reference

Classes

- class [__gnu_cxx::malloc_allocator< _Tp >](#)
An allocator that uses malloc.
This is precisely the allocator defined in the C++ Standard.
 - all allocation calls malloc
 - all deallocation calls free.

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

6.192.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [malloc_allocator.h](#).

6.193 map File Reference

6.193.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [map](#).

6.194 map File Reference

6.194.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map](#).

6.195 map File Reference

6.195.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map](#).

6.196 map.h File Reference

Classes

- class [std::__debug::map](#)< [_Key](#), [_Tp](#), [_Compare](#), [_Allocator](#) >
Class [std::map](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator`
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

6.196.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map.h](#).

6.197 map.h File Reference

Classes

- class [std::__profile::map< _Key, _Tp, _Compare, _Allocator >](#)
Class [std::map](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator`
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator`
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator<= (const map< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator== (const map< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator> (const map< _Key, _Tp, _Compare, _Allocator`
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator>= (const map< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void std::__profile::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

6.197.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map.h](#).

6.198 mask_array.h File Reference

Classes

- class [std::mask_array< _Tp >](#)
Reference to selected subset of an array.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

6.198.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [mask_array.h](#).

6.199 memory File Reference

6.199.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [memory](#).

6.200 memory File Reference

Classes

- struct [__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>](#)

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter >
pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n (_
InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >
pair< _RandomAccessIter, _ForwardIter > __gnu_cxx::__uninitialized_-
copy_n (_RandomAccessIter __first, _Size __count, _ForwardIter __result,
std::random_access_iterator_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >
pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n (_
InputIter __first, _Size __count, _ForwardIter __result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >
pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a (_
InputIter __first, _Size __count, _ForwardIter __result, std::allocator<_Tp>)`

- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a (_-`
`InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter`
`__first, _Size __count, _ForwardIter __result)`

6.200.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/memory](#).

6.201 `merge.h` File Reference

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _-`
`DifferenceTp, typename _Compare >`
`_OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _-`
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __-`
`target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`
`_DifferenceTp, typename _Compare >`
`_OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &__-`
`begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _-`
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`
`_DifferenceTp, typename _Compare >`
`_OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &__-`
`begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _-`
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1 , typename _RAIter3 , typename _Compare >`
`_RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _-`
`RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target,`
`typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _-`
`Compare __comp)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare >`
`_RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _-`
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target,`
`typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _-`
`Compare __comp)`

6.201.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [merge.h](#).

6.202 messages_members.h File Reference

Namespaces

- namespace [std](#)

6.202.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [messages_members.h](#).

6.203 move.h File Reference

Classes

- struct [std::identity< _Tp >](#)
identity

Namespaces

- namespace [std](#)

Defines

- `#define _GLIBCXX_FORWARD(_Tp, __val)`
- `#define _GLIBCXX_MOVE(_Tp)`

Functions

- `template<typename _Tp >`
`_Tp * std::__addressof (_Tp &__r)`
- `template<typename _Tp >`
`_Tp * std::addressof (_Tp &__r)`
- `template<typename _Tp >`
`enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type std::forward`
`(typename std::identity< _Tp >::type &__t)`
- `template<typename _Tp >`
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type std::forward`
`(typename std::identity< _Tp >::type __t)`
- `template<typename _Tp >`
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type std::forward`
`(typename std::remove_reference< _Tp >::type &&__t)`
- `template<typename _Tp >`
`enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type std::forward`
`(typename std::identity< _Tp >::type &&__t)`
- `template<typename _Tp >`
`std::remove_reference< _Tp >::type && std::move (_Tp &&__t)`
- `template<typename _Tp, size_t _Nm>`
`void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b)`

6.203.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [move.h](#).

6.204 mt_allocator.h File Reference

Classes

- struct `__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >`
Policy for shared __pool objects.
- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`
This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.
- class `__gnu_cxx::__mt_alloc_base< _Tp >`
Base class for _Tp dependent member functions.
- struct `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`
Policy for individual __pool objects.
- class `__gnu_cxx::__pool< false >`
Specialization for single thread.
- class `__gnu_cxx::__pool< true >`
Specialization for thread enabled, via gthreads.h.
- struct `__gnu_cxx::__pool_base`
Base class for pool object.

Namespaces

- namespace `__gnu_cxx`

Defines

- `#define __thread_default`

Typedefs

- typedef void(* `__gnu_cxx::__destroy_handler`)(void *)

Functions

- `template<typename _Tp, typename _Poolp >`
`bool __gnu_cxx::operator!= (const __mt_alloc< _Tp, _Poolp > &, const __-`
`mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp, typename _Poolp >`
`bool __gnu_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __-`
`mt_alloc< _Tp, _Poolp > &)`

6.204.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [mt_allocator.h](#).

6.205 multimap.h File Reference

Classes

- class [std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >](#)
Class [std::multimap](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &_`
`lhs, multimap< _Key, _Tp, _Compare, _Allocator > &rhs)`

6.205.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multimap.h](#).

6.206 `multimap.h` File Reference

Classes

- class [std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >](#)
Class [std::__multimap](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator!= (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`
`rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator< (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`_rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator<= (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`_rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator== (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`_rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator> (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`_rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator>= (const multimap< _Key, _Tp, _Compare, _-`
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`_rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void std::__profile::swap (multimap< _Key, _Tp, _Compare, _Allocator >`
`&__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

6.206.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multimap.h](#).

6.207 multiseq_selection.h File Reference

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

Classes

- class [__gnu_parallel::Lexicographic< _T1, _T2, _Compare >](#)
Compare __a pair of types lexicographically, ascending.
- class [__gnu_parallel::LexicographicReverse< _T1, _T2, _Compare >](#)
Compare __a pair of types lexicographically, descending.

Namespaces

- namespace [__gnu_parallel](#)

Defines

- `#define __S(__i)`
- `#define __S(__i)`

Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`
`void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`
`_Tp __gnu_parallel::multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())`

6.207.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets. The algorithm description can be found in P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiseq_selection.h](#).

6.208 multiset.h File Reference

Classes

- class [std::__debug::multiset< _Key, _Compare, _Allocator >](#)
Class [std::multiset](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key , typename _Compare , typename _Allocator >
bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator
> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >
bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator
> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >
bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator
> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >
bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator
> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >
bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator
> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >
bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator
> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >
void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x,
multiset< _Key, _Compare, _Allocator > &__y)`

6.208.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multiset.h](#).

6.209 multiset.h File Reference

Classes

- class [std::__profile::multiset< _Key, _Compare, _Allocator >](#)
Class [std::multiset](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

6.209.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multiset.h](#).

6.210 multiway_merge.h File Reference

Implementation of sequential and parallel multiway merge.

Classes

- struct [__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __-sentinels, _RAIter3, _DifferenceTp, _Compare >](#)

Switch for 3-way merging with `__sentinels` turned off.

- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned on.

- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned off.

- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned on.

- `struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with `__sentinels` turned on.

- `struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with `__sentinels` turned off.

- `class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >`

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

- `struct __gnu_parallel::_LoserTreeTraits< _Tp >`

Traits for determining whether the loser tree should use pointers or copies.

- `struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor.

- `struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-__stable sorting functor.

Namespaces

- namespace `__gnu_parallel`

Defines

- `#define _GLIBCXX_PARALLEL_DECISION(__a, __b, __c, __d)`
- `#define _GLIBCXX_PARALLEL_LENGTH(__s)`
- `#define _GLIBCXX_PARALLEL_MERGE_3_CASE(__a, __b, __c, __c0, __c1)`
- `#define _GLIBCXX_PARALLEL_MERGE_4_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

Functions

- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 __gnu_parallel::__sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`

- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator ,
typename _RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator __-
seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp
__length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator ,
typename _RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator __-
seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp
__length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _-
DifferenceType >
void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __-
seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _-
DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _-
DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp
, typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __-
seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp _-
__length, _Compare __comp)`
- `template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 ,
typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_-
RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-
target, const typename std::iterator_traits< typename std::iterator_traits<
_RAIterIterator >::value_type::first_type >::value_type &__sentinel, _-
DifferenceTp __length, _Compare __comp)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _-
DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_-
RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-
target, const typename std::iterator_traits< typename std::iterator_traits<
_RAIterIterator >::value_type::first_type >::value_type &__sentinel, _-
DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _-
DifferenceType >
void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __-
seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _-
DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _-
DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-
name _Compare >
_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-`

DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, type-`
`name _DifferenceTp, typename _Splitter, typename _Compare >`
`_RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs _`
`begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _`
`DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`
`seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`
`name _Compare >`

```

_RAlterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _
_seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _
DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

```

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _`
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`
`DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`
`__target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`
`__target, _DifferenceTp __length, _Compare __comp, default_parallel_tag`
`__tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`
`__target, _DifferenceTp __length, _Compare __comp, parallel_tag __`
`tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`
`__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact-`
`tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _`
`RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu-`
`parallel::sequential_tag)`

6.210.1 Detailed Description

Implementation of sequential and parallel multiway merge. Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway_merge.h](#).

6.210.2 Define Documentation

6.210.2.1 #define _GLIBCXX_PARALLEL_LENGTH(__s)

Length of a sequence described by a pair of iterators.

Definition at line 53 of file multiway_merge.h.

Referenced by `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

6.211 multiway_mergesort.h File Reference

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::_Piece< _DifferenceTp >](#)
Subsequence description.
- struct [__gnu_parallel::_PMWMSSortingData< _RAIter >](#)
Data accessed by all threads.
- struct [__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >](#)
Split consistently.
- struct [__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >](#)
Split by sampling.
- struct [__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >](#)

Split by exact splitting.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _DifferenceTp >
void __gnu_parallel::__determine_samples (_PMWMSSortingData< _RAIter
> * __sd, _DifferenceTp __num_samples)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >
void __gnu_parallel::parallel_sort_mwms (_RAIter __begin, _RAIter __end, _-
Compare __comp, _ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >
void __gnu_parallel::parallel_sort_mwms_pu (_PMWMSSortingData< _-
RAIter > * __sd, _Compare & __comp)`

6.211.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway_mergesort.h](#).

6.212 mutex File Reference

Classes

- struct [std::adopt_lock_t](#)
Assume the calling thread has already obtained mutex ownership /// and manage it.
- struct [std::defer_lock_t](#)
Do not acquire ownership of the mutex.
- class [std::lock_guard< _Mutex >](#)
Scoped lock idiom.
- class [std::mutex](#)
mutex

- struct `std::once_flag`
once_flag
- class `std::recursive_mutex`
recursive_mutex
- class `std::recursive_timed_mutex`
recursive_timed_mutex
- class `std::timed_mutex`
timed_mutex
- struct `std::try_to_lock_t`
Try to acquire ownership of the mutex without blocking.
- class `std::unique_lock< _Mutex >`
unique_lock

Namespaces

- namespace `std`

Functions

- mutex & `std::__get_once_mutex ()`
- void `std::__once_proxy ()`
- void `std::__set_once_functor_lock_ptr (unique_lock< mutex > *)`
- template<typename _Callable, typename... _Args>
void `std::call_once` (once_flag &__once, _Callable __f, _Args &&...__args)
- template<typename _L1, typename _L2, typename... _L3>
void `std::lock` (_L1 &, _L2 &, _L3 &...)
- template<typename _Mutex >
void `std::swap` (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)
- template<typename _Lock1, typename _Lock2, typename... _Lock3>
int `std::try_lock` (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)

Variables

- function< void()> **std::__once_functor**
- const adopt_lock_t **std::adopt_lock**
- const defer_lock_t **std::defer_lock**
- const try_to_lock_t **std::try_to_lock**

6.212.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [mutex](#).

6.213 nested_exception.h File Reference

Classes

- class [std::nested_exception](#)
Exception class with exception_ptr data member.

Namespaces

- namespace [std](#)

Functions

- template<typename _Ex >
const nested_exception * **std::__get_nested_exception** (const _Ex &__ex)
- template<typename _Ex >
void **std::__throw_with_nested** (_Ex &&, const nested_exception *__ex) __attribute__((__noreturn__))
- template<typename _Ex >
void **std::__throw_with_nested** (_Ex &&, ...) __attribute__((__noreturn__))
- void [std::rethrow_if_nested](#) (const nested_exception &__ex)
- template<typename _Ex >
void [std::rethrow_if_nested](#) (const _Ex &__ex)
- template<typename _Ex >
void [std::throw_with_nested](#) (_Ex __ex)

6.213.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [nested_exception.h](#).

6.214 new File Reference

Classes

- class [std::bad_alloc](#)

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Namespaces

- namespace [std](#)

Typedefs

- typedef void(* [std::new_handler](#))()

Functions

- new_handler [std::set_new_handler](#) (new_handler) throw ()
- void * [operator new](#) (std::size_t) throw (std::bad_alloc)
- void * [operator new\[\]](#) (std::size_t) throw (std::bad_alloc)
- void [operator delete](#) (void *) throw ()
- void [operator delete\[\]](#) (void *) throw ()
- void * [operator new](#) (std::size_t, const std::nothrow_t &) throw ()
- void * [operator new\[\]](#) (std::size_t, const std::nothrow_t &) throw ()
- void [operator delete](#) (void *, const std::nothrow_t &) throw ()
- void [operator delete\[\]](#) (void *, const std::nothrow_t &) throw ()
- void * [operator new](#) (std::size_t, void *__p) throw ()
- void * [operator new\[\]](#) (std::size_t, void *__p) throw ()
- void [operator delete](#) (void *, void *) throw ()
- void [operator delete\[\]](#) (void *, void *) throw ()

Variables

- `const nothrow_t std::nothrow`

6.214.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4 for more.

Definition in file [new](#).

6.214.2 Function Documentation

6.214.2.1 `void operator delete (void *) throw ()`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.2 `void operator delete (void *, const std::nothrow_t &) throw ()`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.3 void operator delete (void *, void *) throw () [inline]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 107 of file `new`.

6.214.2.4 void operator delete[] (void *) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.5 void operator delete[] (void *, const std::nothrow_t &) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.6 void operator delete[] (void *, void *) throw () [inline]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 108 of file `new`.

6.214.2.7 void* operator new (std::size_t, void * __p) throw () [inline]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 103 of file `new`.

6.214.2.8 void* operator new (std::size_t, const std::nothrow_t &) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.9 void* operator new (std::size_t) throw (std::bad_alloc)

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.10 void* operator new[] (std::size_t, const std::nothrow_t &) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.11 void* operator new[] (std::size_t) throw (std::bad_alloc)

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.214.2.12 `void* operator new[] (std::size_t, void * __p) throw ()` `[inline]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 104 of file `new`.

6.215 `new_allocator.h` File Reference

Classes

- class `__gnu_cxx::new_allocator<_Tp>`
*An allocator that uses global new, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls `operator new`
 - all deallocation calls `operator delete`.

Namespaces

- namespace `__gnu_cxx`

Functions

- `template<typename _Tp>`
`bool __gnu_cxx::operator!= (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`
- `template<typename _Tp>`
`bool __gnu_cxx::operator== (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`

6.215.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [new_allocator.h](#).

6.216 numeric File Reference

6.216.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [numeric](#).

6.217 numeric File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >
_Tp __gnu_cxx::__power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >
_Tp __gnu_cxx::__power (_Tp __x, _Integer __n)`
- `template<typename _ForwardIter, typename _Tp >
void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >
_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >
_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`

6.217.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/numeric](#).

6.218 numeric File Reference

Parallel STL function calls corresponding to [stl_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp`
`__init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp`
`__init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::__accumulate_switch (__RAIter __begin, __RAIter __`
`__end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_-`
`tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_-`
`unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __`
`begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _`
`IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __`
`begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,`
`random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__`
`Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1,`
`typename _BinaryFunction2 >`
`_Tp std::__parallel::__inner_product_switch (_RAIter1 __first1, _RAIter1`
`__last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1,`
`_BinaryFunction2 __binary_op2, random_access_iterator_tag, random_-`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_-`
`parallel::parallel_unbalanced)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Tp std::__parallel::__inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`

- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`

6.218.1 Detailed Description

Parallel STL function calls corresponding to [stl_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/numeric](#).

6.219 `numeric_traits.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

Defines

- `#define __glibcxx_digits(_Tp)`
- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max(_Tp)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __glibcxx_min(_Tp)`
- `#define __glibcxx_signed(_Tp)`

6.219.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [numeric_traits.h](#).

6.220 `numericfwd.h` File Reference

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- `template<typename _Iter, typename _Tp, typename _Tag >`
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper,`
`_Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _-`
`BinaryOper, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`
`_Tag2 >`
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-`
`BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-`
`BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu -`
`parallel::Parallelism _parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1,`
`typename BinaryFunction2 >`
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _-`
`RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator _-`
`tag, random_access_iterator_tag, __gnu_parallel::Parallelism=__gnu _-`
`parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`
`name _BinaryFunction2, typename _Tag1, typename _Tag2 >`
`_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp,`
`_BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`
`_Tag2 >`
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _-`
`BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _-`
`BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, __ -`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu -`
`parallel::Parallelism)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init)`

- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`

- `template<typename _Iter1 , typename _Iter2 , typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __`
`__first2, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter , typename _OIter , typename _BinaryOper >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter , typename _OIter >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter , typename _OIter , typename _BinaryOper >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter , typename _OIter >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, __gnu_`
`parallel::sequential_tag)`

6.220.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [numericfwd.h](#).

6.221 omp_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop.
This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result`
`>`
`_Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter`
`__begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _`
`Result &__output, typename std::iterator_traits< _RAIter >::difference_type _`
`__bound)`

6.221.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop.
This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp_loop.h](#).

6.222 omp_loop_static.h File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static
(_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _-
Result __base, _Result &__output, typename std::iterator_traits< _RAIter
>::difference_type __bound)`

6.222.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp_loop_static.h](#).

6.223 os_defines.h File Reference

Defines

- `#define __NO_CTYPE`

6.223.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [os_defines.h](#).

6.224 ostream File Reference

Classes

- class `std::basic_ostream< _CharT, _Traits >`
Controlling output.
This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.
- class `std::basic_ostream< _CharT, _Traits >::sentry`
Performs setup work for output streams.

Namespaces

- namespace `std`

Functions

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::endl (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::ends (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::flush (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &&__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &&__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &&__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &&__out, char __c)`

- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`
`_Traits > &__out, signed char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`
`_Traits > &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`
`_Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`
`_Traits > &__out, const signed char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`
`_Traits > &__out, const unsigned char *__s)`

6.224.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ostream](#).

6.225 ostream.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__out, const char *__s)`

6.225.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream.tcc](#).

6.226 ostream_insert.h File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >
void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template wostream & std::__ostream_insert (wostream &, const wchar_t *, streamsize)`
- `template ostream & std::__ostream_insert (ostream &, const char *, streamsize)`
- `template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >
void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`

6.226.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream_insert.h](#).

6.227 par_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result
>
_Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __-
begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _-
Result &__output, typename std::iterator_traits<_RAIter>::difference_type __-
bound)`

6.227.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [par_loop.h](#).

6.228 parallel.h File Reference

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

6.228.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel.h](#).

6.229 partial_sum.h File Reference

Parallel implementation of [std::partial_sum\(\)](#), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum (_Iter __begin, _Iter
__end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_Iter __-
begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, type-
name std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_Iter __begin,
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename
std::iterator_traits< _Iter >::difference_type __n)`

6.229.1 Detailed Description

Parallel implementation of [std::partial_sum\(\)](#), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partial_sum.h](#).

6.230 partition.h File Reference

Parallel implementation of [std::partition\(\)](#), [std::nth_element\(\)](#), and [std::partial_sort\(\)](#). This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Defines

- `#define _GLIBCXX_VOLATILE`

Functions

- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth,
_RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __-
middle, _RAIter __end, _Compare __comp)`

- `template<typename _RAIter, typename _Predicate >
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_
partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex
__num_threads)`

6.230.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partition.h](#).

6.230.2 Define Documentation

6.230.2.1 #define _GLIBCXX_VOLATILE

Decide whether to declare certain variables volatile.

Definition at line 43 of file [partition.h](#).

Referenced by `__gnu_parallel::__parallel_partition()`.

6.231 pod_char_traits.h File Reference

Classes

- struct [__gnu_cxx::character< V, I, S >](#)
A POD class that serves as a character abstraction class.
- struct [std::char_traits< __gnu_cxx::character< V, I, S > >](#)
char_traits<__gnu_cxx::character> specialization.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<typename V, typename I, typename S >
bool __gnu_cxx::operator< (const character< V, I, S > &lhs, const character<
V, I, S > &rhs)`

- `template<typename V, typename I, typename S >`
`bool __gnu_cxx::operator== (const character< V, I, S > &lhs, const`
`character< V, I, S > &rhs)`

6.231.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pod_char_traits.h](#).

6.232 pointer.h File Reference

Classes

- struct [__gnu_cxx::Invalid_type](#)
- class [__gnu_cxx::Pointer_adapter<_Storage_policy >](#)
- class [__gnu_cxx::Relative_pointer_impl<_Tp >](#)
A storage policy for use with [_Pointer_adapter<>](#) which stores the pointer's address as an offset value which is relative to its own address.
- class [__gnu_cxx::Relative_pointer_impl<const _Tp >](#)
- class [__gnu_cxx::Std_pointer_impl<_Tp >](#)
A storage policy for use with [_Pointer_adapter<>](#) which yields a standard pointer.
- struct [__gnu_cxx::Unqualified_type<_Tp >](#)

Namespaces

- namespace [__gnu_cxx](#)

Defines

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

Functions

- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter<_Tp1 > &__lhs, _Tp2`
`__rhs)`

- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`
`&__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __-`
`rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__-`
`rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const`
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const`
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`
`&__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const`
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`
`__rhs)`
- `template<typename _CharT , typename _Traits , typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<`
`(std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter<`
`_StoreT > &__p)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`
`__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`
`&__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const`
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const`
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _-`
`Pointer_adapter< _Tp > &__rhs)`

- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`
`&__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __-`
`rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__-`
`rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`
`__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const`
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const`
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _-`
`Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`
`&__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`
`__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`
`__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const`
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`
`&__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const`
`_Pointer_adapter< _Tp2 > &__rhs)`

6.232.1 Detailed Description

Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

Definition in file [pointer.h](#).

6.233 pool_allocator.h File Reference

Classes

- class [__gnu_cxx::__pool_alloc<_Tp>](#)
Allocator using a memory pool with a single lock.
- class [__gnu_cxx::__pool_alloc_base](#)
Base class for [__pool_alloc](#).

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Tp>`
`bool __gnu_cxx::operator!= (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`
- `template<typename _Tp>`
`bool __gnu_cxx::operator== (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`

6.233.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pool_allocator.h](#).

6.234 postypes.h File Reference

Classes

- class [std::fpos<_StateT>](#)

Class representing stream positions.

Namespaces

- namespace [std](#)

Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate_t > [std::streampos](#)
- typedef ptrdiff_t [std::streamsize](#)
- typedef fpos< mbstate_t > [std::u16streampos](#)
- typedef fpos< mbstate_t > [std::u32streampos](#)
- typedef fpos< mbstate_t > [std::wstreampos](#)

Functions

- template<typename _StateT >
bool [std::operator!=](#) (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _StateT >
bool [std::operator==](#) (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)

6.234.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [postypes.h](#).

6.235 [priority_queue.hpp](#) File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.235.1 Detailed Description

Contains `priority_queues`.

Definition in file [priority_queue.hpp](#).

6.236 `priority_queue_base_dispatch.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.236.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file [priority_queue_base_dispatch.hpp](#).

6.237 `profiler.h` File Reference

Interface of the profiling runtime library.

Classes

- struct [__gnu_profile::__reentrance_guard](#)
Reentrance guard.

Namespaces

- namespace [__gnu_profile](#)

Defines

- `#define __profcxx_hashtable_construct(__x...)`
- `#define __profcxx_hashtable_construct2(__x...)`
- `#define __profcxx_hashtable_destruct(__x...)`
- `#define __profcxx_hashtable_destruct2(__x...)`

- #define **__profcxx_hashtable_resize**(__x...)
- #define **__profcxx_is_invalid**()
- #define **__profcxx_is_off**()
- #define **__profcxx_is_on**()
- #define **__profcxx_list_construct**(__x...)
- #define **__profcxx_list_construct2**(__x...)
- #define **__profcxx_list_destruct**(__x...)
- #define **__profcxx_list_destruct2**(__x...)
- #define **__profcxx_list_insert**(__x...)
- #define **__profcxx_list_invalid_operator**(__x...)
- #define **__profcxx_list_iterate**(__x...)
- #define **__profcxx_list_operation**(__x...)
- #define **__profcxx_list_rewind**(__x...)
- #define **__profcxx_map_to_unordered_map_construct**(__x...)
- #define **__profcxx_map_to_unordered_map_destruct**(__x...)
- #define **__profcxx_map_to_unordered_map_erase**(__x...)
- #define **__profcxx_map_to_unordered_map_find**(__x...)
- #define **__profcxx_map_to_unordered_map_insert**(__x...)
- #define **__profcxx_map_to_unordered_map_invalidate**(__x...)
- #define **__profcxx_map_to_unordered_map_iterate**(__x...)
- #define **__profcxx_report**()
- #define **__profcxx_turn_off**()
- #define **__profcxx_turn_on**()
- #define **__profcxx_vector_construct**(__x...)
- #define **__profcxx_vector_construct2**(__x...)
- #define **__profcxx_vector_destruct**(__x...)
- #define **__profcxx_vector_destruct2**(__x...)
- #define **__profcxx_vector_find**(__x...)
- #define **__profcxx_vector_insert**(__x...)
- #define **__profcxx_vector_invalid_operator**(__x...)
- #define **__profcxx_vector_iterate**(__x...)
- #define **__profcxx_vector_resize**(__x...)
- #define **__profcxx_vector_resize2**(__x...)
- #define **_GLIBCXX_PROFILE_DATA**(__name)
- #define **_GLIBCXX_PROFILE_DEFINE_DATA**(__type, __name, __initial_value...)
- #define **_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA**(__type, __name)
- #define **_GLIBCXX_PROFILE_MAX_STACK_DEPTH**
- #define **_GLIBCXX_PROFILE_MAX_STACK_DEPTH_ENV_VAR**
- #define **_GLIBCXX_PROFILE_MAX_WARN_COUNT**
- #define **_GLIBCXX_PROFILE_MAX_WARN_COUNT_ENV_VAR**
- #define **_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC**
- #define **_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR**

- `#define _GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__trace_hash_func_construct (const void *)`
- `void __gnu_profile::__trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_construct (const void *, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_construct (const void *)`
- `void __gnu_profile::__trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_set_find (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_set_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_operation (const void *)`
- `void __gnu_profile::__trace_list_to_slist_rewind (const void *)`
- `void __gnu_profile::__trace_list_to_vector_construct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_construct (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_destruct (const void *)`

- void `__gnu_profile::__trace_map_to_unordered_map_erase` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_find` (const void *, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void *)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_size_construct` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_size_resize` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_construct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_find` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void *)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void *, std::size_t, std::size_t)
- bool `__gnu_profile::__turn_off` ()
- bool `__gnu_profile::__turn_on` ()

6.237.1 Detailed Description

Interface of the profiling runtime library.

Definition in file [profiler.h](#).

6.238 profiler_algos.h File Reference

Algorithms used by the profile extension.

Namespaces

- namespace [__gnu_profile](#)

Functions

- `template<typename _InputIterator, typename _Function>`
`_Function __gnu_profile::__for_each (_InputIterator __first, _InputIterator __-`
`last, _Function __f)`
- `template<typename _Container>`
`void __gnu_profile::__insert_top_n (_Container &__output, const typename`
`_Container::value_type &__value, typename _Container::size_type __n)`
- `template<typename _ForwardIterator, typename _Tp>`
`_ForwardIterator __gnu_profile::__remove (_ForwardIterator __first, _-`
`ForwardIterator __last, const _Tp &__value)`
- `template<typename _Container>`
`void __gnu_profile::__top_n (const _Container &__input, _Container &__-`
`output, typename _Container::size_type __n)`

6.238.1 Detailed Description

Algorithms used by the profile extension. This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

Definition in file [profiler_algos.h](#).

6.239 profiler_hashtable_size.h File Reference

Collection of hashtable size traces.

Classes

- class [__gnu_profile::__trace_hashtable_size](#)
Hashtable size instrumentation trace producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- `void __gnu_profile::__trace_hashtable_size_construct (const void *,`
`std::size_t)`

- void `__gnu_profile::__trace_hashtable_size_destruct` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_hashtable_size_init` ()
- void `__gnu_profile::__trace_hashtable_size_report` (FILE *__f, __warning_vector_t &__warnings)
- void `__gnu_profile::__trace_hashtable_size_resize` (const void *, std::size_t, std::size_t)

6.239.1 Detailed Description

Collection of hashtable size traces.

Definition in file [profiler_hashtable_size.h](#).

6.240 profiler_list_to_slist.h File Reference

Diagnostics for list to slist.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void `__gnu_profile::__trace_list_to_slist_construct` (const void *)
- void `__gnu_profile::__trace_list_to_slist_destruct` (const void *)
- void `__gnu_profile::__trace_list_to_slist_init` ()
- void `__gnu_profile::__trace_list_to_slist_operation` (const void *)
- void `__gnu_profile::__trace_list_to_slist_report` (FILE *__f, __warning_vector_t &__warnings)
- void `__gnu_profile::__trace_list_to_slist_rewind` (const void *)

6.240.1 Detailed Description

Diagnostics for list to slist.

Definition in file [profiler_list_to_slist.h](#).

6.241 profiler_list_to_vector.h File Reference

diagnostics for list to vector.

Classes

- class [__gnu_profile::__list2vector_info](#)
A list-to-vector instrumentation line in the object table.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_list_to_vector_construct](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_destruct](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_init](#) ()
- void [__gnu_profile::__trace_list_to_vector_insert](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_list_to_vector_invalid_operator](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_iterate](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_list_to_vector_report](#) (FILE *__f, __warning_vector_t &__warnings)
- void [__gnu_profile::__trace_list_to_vector_resize](#) (const void *, std::size_t, std::size_t)

6.241.1 Detailed Description

diagnostics for list to vector.

Definition in file [profiler_list_to_vector.h](#).

6.242 profiler_map_to_unordered_map.h File Reference

Diagnostics for map to unordered_map.

Classes

- class [__gnu_profile::__map2umap_info](#)
A map-to-unordered_map instrumentation line in the object table.

- class [__gnu_profile::__map2umap_stack_info](#)
A map-to-unordered_map instrumentation line in the stack table.
- class [__gnu_profile::__trace_map2umap](#)
Map-to-unordered_map instrumentation producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- int [__gnu_profile::__log2](#) (std::size_t __size)
- float [__gnu_profile::__map_erase_cost](#) (std::size_t __size)
- float [__gnu_profile::__map_find_cost](#) (std::size_t __size)
- float [__gnu_profile::__map_insert_cost](#) (std::size_t __size)
- void [__gnu_profile::__trace_map_to_unordered_map_construct](#) (const void *)
- void [__gnu_profile::__trace_map_to_unordered_map_destruct](#) (const void *)
- void [__gnu_profile::__trace_map_to_unordered_map_erase](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_find](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_init](#) ()
- void [__gnu_profile::__trace_map_to_unordered_map_insert](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_invalidate](#) (const void *)
- void [__gnu_profile::__trace_map_to_unordered_map_iterate](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_report](#) (FILE *__f, __warning_vector_t &__warnings)

6.242.1 Detailed Description

Diagnostics for map to unordered_map.

Definition in file [profiler_map_to_unordered_map.h](#).

6.243 profiler_node.h File Reference

Data structures to represent a single profiling event.

Classes

- class [__gnu_profile::__object_info_base](#)
Base class for a line in the object table.
- class [__gnu_profile::__stack_hash](#)
Hash function for summary trace using call stack as index.
- class [__gnu_profile::__stack_info_base< __object_info >](#)
Base class for a line in the stack table.

Namespaces

- namespace [__gnu_profile](#)

Typedefs

- typedef void * [__gnu_profile::__instruction_address_t](#)
- typedef const void * [__gnu_profile::__object_t](#)
- typedef std::vector< [__instruction_address_t](#) > [__gnu_profile::__stack_npt](#)
- typedef [__stack_npt](#) * [__gnu_profile::__stack_t](#)

Functions

- [__stack_t __gnu_profile::__get_stack \(\)](#)
- [std::size_t __gnu_profile::__size \(__stack_t __stack\)](#)
- [std::size_t __gnu_profile::__stack_max_depth \(\)](#)
- [void __gnu_profile::__write \(FILE *__f, __stack_t __stack\)](#)

6.243.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file [profiler_node.h](#).

6.244 profiler_state.h File Reference

Global profiler state.

Namespaces

- namespace [__gnu_profile](#)

Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

Functions

- bool `__gnu_profile::__is_invalid` ()
- bool `__gnu_profile::__is_off` ()
- bool `__gnu_profile::__is_on` ()
- bool `__gnu_profile::__turn` (`__state_type` __s)
- bool `__gnu_profile::__turn_off` ()
- bool `__gnu_profile::__turn_on` ()
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__state_type`, `__state`, `__INVALID`)

6.244.1 Detailed Description

Global profiler state.

Definition in file [profiler_state.h](#).

6.245 profiler_trace.h File Reference

Diagnostics for container sizes.

Classes

- class [__gnu_profile::__trace_base](#)< [__object_info](#), [__stack_info](#) >
Base class for all trace producers.
- struct [__gnu_profile::__warning_data](#)
Representation of a warning.

Namespaces

- namespace [__gnu_profile](#)

Defines

- `#define _GLIBCXX_IMPL_UNORDERED_MAP`

Typedefs

- `typedef std::vector< __cost_factor * > __gnu_profile::__cost_factor_vector`
- `typedef std::unordered_map< std::string, std::string > __gnu_profile::__env_t`
- `typedef std::vector< __warning_data > __gnu_profile::__warning_vector_t`

Functions

- `std::size_t __gnu_profile::__env_to_size_t (const char *__env_var, std::size_t __default_value)`
- `__cost_factor_vector * __gnu_profile::__get__cost_factors ()`
- `__env_t & __gnu_profile::__get__env ()`
- `__gnu_cxx::mutex & __gnu_profile::__get__global_lock ()`
- `__cost_factor & __gnu_profile::__get__list_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__list_resize_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__list_shift_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_erase_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_find_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_insert_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_erase_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_find_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_insert_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_resize_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_shift_cost_factor ()`
- `__trace_hash_func * __gnu_profile::__get__S_hash_func ()`
- `__trace_hashtable_size * __gnu_profile::__get__S_hashtable_size ()`
- `__trace_list_to_slist * __gnu_profile::__get__S_list_to_slist ()`
- `__trace_list_to_vector * __gnu_profile::__get__S_list_to_vector ()`

- `__trace_map2umap *& __gnu_profile::__get__S_map2umap ()`
- `std::size_t & __gnu_profile::__get__S_max_mem ()`
- `std::size_t & __gnu_profile::__get__S_max_stack_depth ()`
- `std::size_t & __gnu_profile::__get__S_max_warn_count ()`
- `const char *& __gnu_profile::__get__S_trace_file_name ()`
- `__trace_vector_size *& __gnu_profile::__get__S_vector_size ()`
- `__trace_vector_to_list *& __gnu_profile::__get__S_vector_to_list ()`
- `int __gnu_profile::__log_magnitude (float __f)`
- `std::size_t __gnu_profile::__max_mem ()`
- `FILE * __gnu_profile::__open_output_file (const char *__extension)`
- `bool __gnu_profile::__profcxx_init ()`
- `void __gnu_profile::__profcxx_init_unconditional ()`
- `void __gnu_profile::__read_cost_factors ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__set_cost_factors ()`
- `void __gnu_profile::__set_max_mem ()`
- `void __gnu_profile::__set_max_stack_trace_depth ()`
- `void __gnu_profile::__set_max_warn_count ()`
- `void __gnu_profile::__set_trace_path ()`
- `std::size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__trace_hash_func_init ()`
- `void __gnu_profile::__trace_hash_func_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_hashtable_size_init ()`
- `void __gnu_profile::__trace_hashtable_size_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_slist_init ()`
- `void __gnu_profile::__trace_list_to_slist_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_vector_init ()`
- `void __gnu_profile::__trace_list_to_vector_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_map_to_unordered_map_init ()`
- `void __gnu_profile::__trace_map_to_unordered_map_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_vector_size_init ()`
- `void __gnu_profile::__trace_vector_size_report (FILE *, __warning_vector_t &)`
- `void __gnu_profile::__trace_vector_to_list_init ()`
- `void __gnu_profile::__trace_vector_to_list_report (FILE *, __warning_vector_t &)`
- `void __gnu_profile::__write_cost_factors ()`

6.245.1 Detailed Description

Diagnostics for container sizes. Data structures to represent profiling traces.

Definition in file [profiler_trace.h](#).

6.246 profiler_vector_size.h File Reference

Collection of vector size traces.

Classes

- class [__gnu_profile::__trace_vector_size](#)
Hashtable size instrumentation trace producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_vector_size_construct](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_vector_size_destruct](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_vector_size_init](#) ()
- void [__gnu_profile::__trace_vector_size_report](#) (FILE *, __warning_vector_t &)
- void [__gnu_profile::__trace_vector_size_resize](#) (const void *, std::size_t, std::size_t)

6.246.1 Detailed Description

Collection of vector size traces.

Definition in file [profiler_vector_size.h](#).

6.247 profiler_vector_to_list.h File Reference

diagnostics for vector to list.

Classes

- class [__gnu_profile::__trace_vector_to_list](#)
Vector-to-list instrumentation producer.
- class [__gnu_profile::__vector2list_info](#)
A vector-to-list instrumentation line in the object table.
- class [__gnu_profile::__vector2list_stack_info](#)
A vector-to-list instrumentation line in the stack table.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_vector_to_list_construct](#) (const void *)
- void [__gnu_profile::__trace_vector_to_list_destruct](#) (const void *)
- void [__gnu_profile::__trace_vector_to_list_find](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_vector_to_list_init](#) ()
- void [__gnu_profile::__trace_vector_to_list_insert](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_vector_to_list_invalid_operator](#) (const void *)
- void [__gnu_profile::__trace_vector_to_list_iterate](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_vector_to_list_report](#) (FILE *, __warning_vector_t &)
- void [__gnu_profile::__trace_vector_to_list_resize](#) (const void *, std::size_t, std::size_t)

6.247.1 Detailed Description

diagnostics for vector to list.

Definition in file [profiler_vector_to_list.h](#).

6.248 queue File Reference

6.248.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [queue](#).

6.249 queue.h File Reference

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- class [__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>](#)

Double-ended queue of bounded size, allowing lock-free atomic access. [push_front\(\)](#) and [pop_front\(\)](#) must not be called concurrently to each other, while [pop_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.

Namespaces

- namespace [__gnu_parallel](#)

Defines

- [#define _GLIBCXX_VOLATILE](#)

6.249.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [queue.h](#).

6.249.2 Define Documentation

6.249.2.1 [#define _GLIBCXX_VOLATILE](#)

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file [queue.h](#).

6.250 quicksort.h File Reference

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

6.250.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [quicksort.h](#).

6.251 random File Reference

6.251.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [random](#).

6.252 random.h File Reference

Classes

- class [std::bernoulli_distribution](#)
A Bernoulli random number distribution.
- struct [std::bernoulli_distribution::param_type](#)
- class [std::binomial_distribution<_IntType>](#)
A discrete binomial random number distribution.
- struct [std::binomial_distribution<_IntType>::param_type](#)
- class [std::cauchy_distribution<_RealType>](#)
*A *cauchy_distribution* random number distribution.*
- struct [std::cauchy_distribution<_RealType>::param_type](#)
- class [std::chi_squared_distribution<_RealType>](#)
*A *chi_squared_distribution* random number distribution.*
- struct [std::chi_squared_distribution<_RealType>::param_type](#)
- class [std::discard_block_engine<_RandomNumberEngine, __p, __r>](#)
- class [std::discrete_distribution<_IntType>](#)
*A *discrete_distribution* random number distribution.*
- struct [std::discrete_distribution<_IntType>::param_type](#)
- class [std::exponential_distribution<_RealType>](#)
An exponential continuous distribution for random numbers.
- struct [std::exponential_distribution<_RealType>::param_type](#)
- class [std::extreme_value_distribution<_RealType>](#)
*A *extreme_value_distribution* random number distribution.*
- struct [std::extreme_value_distribution<_RealType>::param_type](#)
- class [std::fisher_f_distribution<_RealType>](#)
*A *fisher_f_distribution* random number distribution.*
- struct [std::fisher_f_distribution<_RealType>::param_type](#)
- class [std::gamma_distribution<_RealType>](#)
A gamma continuous distribution for random numbers.
- struct [std::gamma_distribution<_RealType>::param_type](#)
- class [std::geometric_distribution<_IntType>](#)

A discrete geometric random number distribution.

- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`

A model of a linear congruential random number generator.

- class `std::lognormal_distribution<_RealType>`

A [lognormal_distribution](#) random number distribution.

- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::negative_binomial_distribution<_IntType>`

A [negative_binomial_distribution](#) random number distribution.

- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`

A normal continuous distribution for random numbers.

- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`

A [piecewise_constant_distribution](#) random number distribution.

- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`

A [piecewise_linear_distribution](#) random number distribution.

- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`

A discrete Poisson random number distribution.

- struct `std::poisson_distribution<_IntType>::param_type`
- class `std::random_device`
- class `std::seed_seq`

The [seed_seq](#) class generates sequences of seeds for random number generators.

- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

- class `std::student_t_distribution<_RealType>`

A [student_t_distribution](#) random number distribution.

- struct [std::student_t_distribution<_RealType>::param_type](#)
- class [std::uniform_int_distribution<_IntType>](#)
Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.
- struct [std::uniform_int_distribution<_IntType>::param_type](#)
- class [std::uniform_real_distribution<_RealType>](#)
Uniform continuous distribution for random numbers.
- struct [std::uniform_real_distribution<_RealType>::param_type](#)
- class [std::weibull_distribution<_RealType>](#)
A [weibull_distribution](#) random number distribution.
- struct [std::weibull_distribution<_RealType>::param_type](#)

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Typedefs

- typedef minstd_rand0 [std::default_random_engine](#)
- typedef shuffle_order_engine< minstd_rand0, 256 > [std::knuth_b](#)
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > [std::minstd_rand](#)
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > [std::minstd_rand0](#)
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [std::mt19937_64](#)
- typedef discard_block_engine< ranlux24_base, 223, 23 > [std::ranlux24](#)
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > [std::ranlux24_base](#)
- typedef discard_block_engine< ranlux48_base, 389, 11 > [std::ranlux48](#)
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > [std::ranlux48_base](#)

Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >
_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool std::operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>
bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _IntType >
bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _RealType >
bool std::operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >
bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >
bool std::operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >
bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _IntType >
bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`

- `template<typename _RealType >`
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1,`
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__`
`__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _`
`_UIntType __f>`
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, _`
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine , size_t __w, typename _UIntType >`
`bool std::operator!= (const std::independent_bits_engine< _`
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_`
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RealType >`
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const`
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType >`
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType >`
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1,`
`const std::uniform_real_distribution< _IntType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const`
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1,`
`const std::binomial_distribution< _IntType > &__d2)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)`
- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `bool std::operator== (const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::exponential_distribution< _RealType > & __d1, const std::exponential_distribution< _RealType > & __d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::uniform_int_distribution< _IntType > & __d1, const std::uniform_int_distribution< _IntType > & __d2)`

- `template<typename _IntType >`
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1,`
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::uniform_real_distribution< _IntType > &__d1,`
`const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const`
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1,`
`const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1,`
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1,`
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::piecewise_linear_distribution< _RealType >`
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::piecewise_constant_distribution< _RealType >`
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &__is, std::bernoulli_distribution &__x)`

6.252.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.h](#).

6.253 random.tcc File Reference

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Functions

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__detail::transform (_InputIterator __first, _-`
`InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`
`_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT`
`, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _-`
`UIntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`
`UIntType __f, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _-`
`UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
`&__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &__os, const discard_block_engine< _-`
`RandomNumberEngine, __p, __r > &__x)`

- `template<typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)`

```

ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType >
&__x)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _
RealType > &)
• template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType >
&__x)
• template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename
_Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _
UIntType, __w, __s, __r > &__x)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType
> &)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _
RealType > &__x)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _
RealType > &__x)
• template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType
> &)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _
RealType > &)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType >
&__x)
• template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_
ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType
> &__x)
• template<typename _RealType >
bool std::operator== (const std::normal_distribution< _RealType > &__d1,
const std::normal_distribution< _RealType > &__d2)

```

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &_`
`__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType`
`> &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType,`
`__w, __s, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`
`istream< _CharT, _Traits > &__is, piecewise_linear_distribution< _RealType`
`> &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

6.253.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.tcc](#).

6.254 `random_number.h` File Reference

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- class [__gnu_parallel::_RandomNumber](#)
Random number generator, based on the Mersenne twister.

Namespaces

- namespace [__gnu_parallel](#)

6.254.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random_number.h](#).

6.255 `random_shuffle.h` File Reference

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>](#)
Data known to every thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).
- struct [__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>](#)
Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Namespaces

- namespace [__gnu_parallel](#)

Typedefs

- typedef unsigned short [__gnu_parallel::_BinIndex](#)

Functions

- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle](#) (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle_drs](#) (_RAIter __begin, _RAIter __end, typename [std::iterator_traits<_RAIter>::difference_type](#) __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle_drs_pu](#) (_DRSSorterPU<_RAIter, _RandomNumberGenerator> *__pus)
- template<typename _RandomNumberGenerator>
int [__gnu_parallel::__random_number_pow2](#) (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Tp>
_Tp [__gnu_parallel::__round_up_to_pow2](#) (_Tp __x)
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__sequential_random_shuffle](#) (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)

6.255.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random_shuffle.h](#).

6.256 ratio File Reference

Classes

- struct [std::ratio< _Num, _Den >](#)
Provides compile-time rational arithmetic.
- struct [std::ratio_add< _R1, _R2 >](#)
ratio_add
- struct [std::ratio_divide< _R1, _R2 >](#)
ratio_divide
- struct [std::ratio_equal< _R1, _R2 >](#)
ratio_equal
- struct [std::ratio_greater< _R1, _R2 >](#)
ratio_greater
- struct [std::ratio_greater_equal< _R1, _R2 >](#)
ratio_greater_equal
- struct [std::ratio_less< _R1, _R2 >](#)
ratio_less
- struct [std::ratio_less_equal< _R1, _R2 >](#)
ratio_less_equal
- struct [std::ratio_multiply< _R1, _R2 >](#)
ratio_multiply
- struct [std::ratio_not_equal< _R1, _R2 >](#)
ratio_not_equal
- struct [std::ratio_subtract< _R1, _R2 >](#)
ratio_subtract

Namespaces

- namespace [std](#)

Typedefs

- typedef ratio< 1, 1000000000000000000 > **std::atto**
- typedef ratio< 1, 100 > **std::centi**
- typedef ratio< 10, 1 > **std::deca**
- typedef ratio< 1, 10 > **std::deci**
- typedef ratio< 1000000000000000000, 1 > **std::exa**
- typedef ratio< 1, 1000000000000000 > **std::femto**
- typedef ratio< 1000000000, 1 > **std::giga**
- typedef ratio< 100, 1 > **std::hecto**
- typedef ratio< 1000, 1 > **std::kilo**
- typedef ratio< 1000000, 1 > **std::mega**
- typedef ratio< 1, 1000000 > **std::micro**
- typedef ratio< 1, 1000 > **std::milli**
- typedef ratio< 1, 1000000000 > **std::nano**
- typedef ratio< 1000000000000000000, 1 > **std::peta**
- typedef ratio< 1, 1000000000000 > **std::pico**
- typedef ratio< 10000000000000, 1 > **std::tera**

6.256.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ratio](#).

6.257 rb_tree File Reference

Classes

- struct [__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)

6.257.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rb_tree](#).

6.258 rc_string_base.h File Reference

Classes

- class [__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)

6.258.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [rc_string_base.h](#).

6.259 regex File Reference

6.259.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [regex](#).

6.260 regex.h File Reference

Classes

- class [std::basic_regex<_Ch_type, _Rx_traits>](#)
- class [std::match_results<_Bi_iter, _Allocator>](#)

The results of a match or search operation.

- class [std::regex_iterator](#)< [_Bi_iter](#), [_Ch_type](#), [_Rx_traits](#) >
- class [std::regex_token_iterator](#)< [_Bi_iter](#), [_Ch_type](#), [_Rx_traits](#) >
- struct [std::regex_traits](#)< [_Ch_type](#) >

Describes aspects of a regular expression.

- class [std::sub_match](#)< [_BiIter](#) >

Namespaces

- namespace [std](#)

Typedefs

- typedef [match_results](#)< const char * > **std::cmatch**
- typedef [regex_iterator](#)< const char * > **std::cregex_iterator**
- typedef [regex_token_iterator](#)< const char * > [std::cregex_token_iterator](#)
- typedef [sub_match](#)< const char * > [std::csub_match](#)
- typedef [basic_regex](#)< char > [std::regex](#)
- typedef [match_results](#)< [string::const_iterator](#) > **std::smatch**
- typedef [regex_iterator](#)< [string::const_iterator](#) > **std::sregex_iterator**
- typedef [regex_token_iterator](#)< [string::const_iterator](#) > [std::sregex_token_iterator](#)
- typedef [sub_match](#)< [string::const_iterator](#) > [std::ssub_match](#)
- typedef [match_results](#)< const wchar_t * > **std::wcmatch**
- typedef [regex_iterator](#)< const wchar_t * > **std::wcregex_iterator**
- typedef [regex_token_iterator](#)< const wchar_t * > [std::wcregex_token_iterator](#)
- typedef [sub_match](#)< const wchar_t * > [std::wcsub_match](#)
- typedef [basic_regex](#)< wchar_t > [std::wregex](#)
- typedef [match_results](#)< [wstring::const_iterator](#) > **std::wsmatch**
- typedef [regex_iterator](#)< [wstring::const_iterator](#) > **std::wsregex_iterator**
- typedef [regex_token_iterator](#)< [wstring::const_iterator](#) > [std::wsregex_token_iterator](#)
- typedef [sub_match](#)< [wstring::const_iterator](#) > [std::wssub_match](#)

Functions

- template<typename [_Bi_iter](#) >
const [sub_match](#)< [_Bi_iter](#) > & **std::__unmatched_sub** ()
- template<typename [_BiIter](#) >
bool [std::operator!=](#) (const [sub_match](#)< [_BiIter](#) > &__lhs, const [sub_match](#)< [_BiIter](#) > &__rhs)

- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Allocator >`
`bool std::operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _BiIter >`
`bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`

- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream<`
`_Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const basic_-`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_-`
`alloc > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const basic_-`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_-`
`alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool std::operator== (const match_results< _Bi_iter, _Allocator > &__m1,`
`const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`

- `template<typename _BiIter >`
`bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _`
`_Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`void std::swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results<`
`_Bi_iter, _Allocator > &__rhs)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`
`>`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter,`
`_Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_`
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type _`
`__flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type`
`*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_`
`type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc`
`> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`
`_Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _`
`Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_`
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`
`>`


```

bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, match_results< _-
    Bi_iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits >
    &__re, regex_constants::match_flag_type __flags=regex_constants::match_
    default)
• template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >
    bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, const basic_
    regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type _
    __flags=regex_constants::match_default)
• template<typename _Ch_type, class _Allocator, class _Rx_traits >
    bool std::regex\_search (const _Ch_type *__s, match_results< const _Ch_
    type *, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &_
    __e, regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_type, typename _Rx_traits >
    bool std::regex\_search (const _Ch_type *__s, const basic_regex< _Ch_
    type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
    constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename
    _Rx_traits >
    bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _String_
    allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_
    constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_
    type, typename _Rx_traits >
    bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc
    > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _
    Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _
    Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
    constants::match_default)
• template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type
    >
    _Out_iter std::regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __
    last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string<
    _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_
    constants::match_default)
• template<typename _Rx_traits, typename _Ch_type >
    basic_string< _Ch_type > std::regex\_replace (const basic_string< _Ch_
    type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const
    basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __
    flags=regex_constants::match_default)

```

6.260.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex.h](#).

6.261 `regex_compiler.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InIter, typename _TraitsT >
_AutomatonPtr std::regex::compile(const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)`

6.261.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_compiler.h](#).

6.262 `regex_constants.h` File Reference

Constant definitions for the std regex library.

Namespaces

- namespace [std](#)
- namespace [std::regex_constants](#)

5.1 Regular Expression Syntax Options

- `enum std::regex_constants::__syntax_option {
 _S_icode, _S_nosubs, _S_optimize, _S_collate,
 _S_ECMAScript, _S_basic, _S_extended, _S_awk,
 _S_grep, _S_egrep, _S_syntax_last }`
- `typedef unsigned int std::regex_constants::syntax_option_type`
- `static const syntax_option_type std::regex_constants::icase`
- `static const syntax_option_type std::regex_constants::nosubs`
- `static const syntax_option_type std::regex_constants::optimize`
- `static const syntax_option_type std::regex_constants::collate`

- static const syntax_option_type `std::regex_constants::ECMAScript`
- static const syntax_option_type `std::regex_constants::basic`
- static const syntax_option_type `std::regex_constants::extended`
- static const syntax_option_type `std::regex_constants::awk`
- static const syntax_option_type `std::regex_constants::grep`
- static const syntax_option_type `std::regex_constants::egrep`

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `std::regex_constants::__match_flag` {
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- typedef `std::bitset< _S_match_flag_last > std::regex_constants::match_flag_type`
- static const match_flag_type `std::regex_constants::match_default`
- static const match_flag_type `std::regex_constants::match_not_bol`
- static const match_flag_type `std::regex_constants::match_not_eol`
- static const match_flag_type `std::regex_constants::match_not_bow`
- static const match_flag_type `std::regex_constants::match_not_eow`
- static const match_flag_type `std::regex_constants::match_any`
- static const match_flag_type `std::regex_constants::match_not_null`
- static const match_flag_type `std::regex_constants::match_continuous`
- static const match_flag_type `std::regex_constants::match_prev_avail`
- static const match_flag_type `std::regex_constants::format_default`
- static const match_flag_type `std::regex_constants::format_sed`
- static const match_flag_type `std::regex_constants::format_no_copy`
- static const match_flag_type `std::regex_constants::format_first_only`

6.262.1 Detailed Description

Constant definitions for the `std` regex library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `regex_constants.h`.

6.263 `regex_cursor.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _FwdIterT >
_SpecializedCursor< _FwdIterT > std::__regex::__cursor (const _FwdIterT
&__b, const _FwdIterT __e)`

6.263.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_cursor.h](#).

6.264 `regex_error.h` File Reference

Error and exception objects for the std regex library.

Classes

- class [std::regex_error](#)
*A regular expression exception class.
The regular expression library throws objects of this class on error.*

Namespaces

- namespace [std](#)
- namespace [std::regex_constants](#)

Functions

- `void std::__throw_regex_error (regex_constants::error_type __ecode)`

5.3 Error Types

- enum [std::regex_constants::error_type](#) {
[_S_error_collate](#), [_S_error_ctype](#), [_S_error_escape](#), [_S_error_backref](#),
[_S_error_brack](#), [_S_error_paren](#), [_S_error_brace](#), [_S_error_badbrace](#),
[_S_error_range](#), [_S_error_space](#), [_S_error_badrepeat](#), [_S_error_](#)-
[complexity](#),
[_S_error_stack](#), [_S_error_last](#) }
- static const error_type [std::regex_constants::error_collate](#) ([_S_error_collate](#))
- static const error_type [std::regex_constants::error_ctype](#) ([_S_error_ctype](#))
- static const error_type [std::regex_constants::error_escape](#) ([_S_error_escape](#))
- static const error_type [std::regex_constants::error_backref](#) ([_S_error_backref](#))
- static const error_type [std::regex_constants::error_brack](#) ([_S_error_brack](#))
- static const error_type [std::regex_constants::error_paren](#) ([_S_error_paren](#))
- static const error_type [std::regex_constants::error_brace](#) ([_S_error_brace](#))
- static const error_type [std::regex_constants::error_badbrace](#) ([_S_error_](#)-
[badbrace](#))
- static const error_type [std::regex_constants::error_range](#) ([_S_error_range](#))
- static const error_type [std::regex_constants::error_space](#) ([_S_error_space](#))
- static const error_type [std::regex_constants::error_badrepeat](#) ([_S_error_](#)-
[badrepeat](#))
- static const error_type [std::regex_constants::error_complexity](#) ([_S_error_](#)-
[complexity](#))
- static const error_type [std::regex_constants::error_stack](#) ([_S_error_stack](#))

6.264.1 Detailed Description

Error and exception objects for the std regex library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_error.h](#).

6.265 regex_grep_matcher.h File Reference

Namespaces

- namespace [std](#)

Typedefs

- typedef [std::stack](#)< [_StateIdT](#), [std::vector](#)< [_StateIdT](#) > > [std::__regex::-](#)
[StateStack](#)

6.265.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_grep_matcher.h](#).

6.266 regex_grep_matcher.tcc File Reference

Namespaces

- namespace [std](#)

6.266.1 Detailed Description

Definition in file [regex_grep_matcher.tcc](#).

6.267 regex_nfa.h File Reference

Namespaces

- namespace [std](#)

Typedefs

- typedef [std::shared_ptr](#)< _Automaton > **std::__regex::AutomatonPtr**
- typedef std::function< bool(const _PatternCursor &)> **std::__regex::_Matcher**
- typedef int **std::__regex::StateIdT**
- typedef [std::set](#)< _StateIdT > **std::__regex::StateSet**
- typedef std::function< void(const _PatternCursor &, _Results &)> **std::__regex::_Tagger**

Enumerations

- enum **_Opcode** {
 _S_opcode_unknown, _S_opcode_alternative, _S_opcode_subexpr_begin,
 _S_opcode_subexpr_end,
 _S_opcode_match, _S_opcode_accept }

Functions

- `bool std::__regex::_AnyMatcher (const _PatternCursor &)`

Variables

- `static const _StateIdT std::__regex::_S_invalid_state_id`

6.267.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_nfa.h](#).

6.268 `regex_nfa.tcc` File Reference

Namespaces

- namespace [std](#)

6.268.1 Detailed Description

Definition in file [regex_nfa.tcc](#).

6.269 `rope` File Reference

Classes

- class [__gnu_cxx::rope<_CharT, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [__gnu_cxx::__detail](#)
- namespace [std](#)
- namespace [std::tr1](#)

Defines

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`

Typedefs

- `typedef rope< char > __gnu_cxx::crope`
- `typedef rope< wchar_t > __gnu_cxx::wrope`

Enumerations

- `enum { _S_max_rope_depth }`
- `enum _Tag { _S_leaf, _S_concat, _S_substringfn, _S_function }`

Functions

- `crope::reference __gnu_cxx::mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _ForwardIterator, typename _Tp >
void __gnu_cxx::Destroy_const (_ForwardIterator __first, _ForwardIterator
__last, allocator< _Tp >)`
- `template<typename _ForwardIterator, typename _Allocator >
void __gnu_cxx::Destroy_const (_ForwardIterator __first, _ForwardIterator
__last, _Allocator __alloc)`
- `template<class _CharT >
void __gnu_cxx::_S_cond_store_eos (_CharT &)`
- `void __gnu_cxx::_S_cond_store_eos (char &__c)`
- `void __gnu_cxx::_S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >
_CharT __gnu_cxx::_S_eos (_CharT *)`
- `bool __gnu_cxx::_S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >
bool __gnu_cxx::_S_is_basic_char_type (_CharT *)`
- `bool __gnu_cxx::_S_is_basic_char_type (char *)`
- `template<class _CharT >
bool __gnu_cxx::_S_is_one_byte_char_type (_CharT *)`

- `bool __gnu_cxx::S_is_one_byte_char_type (char *)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const rope< _CharT, _Alloc > &__x, const`
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc`
`> &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_-`
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const`
`_Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _-`
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __-`
`n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, _CharT __right)`

- `template<class _CharT, class _Alloc >`
`ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc`
`> &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_-`
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &_-`
`__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _-`
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const`
`rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<`
`(std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc >`
`&__r)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const`
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc`
`> &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc >`
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

Variables

- `rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

6.269.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rope](#).

6.270 ropeimpl.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<class _CharT, class _Traits >`
`void __gnu_cxx::Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `template<class _Rope_iterator >`
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

6.270.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ropeimpl.h](#).

6.271 safe_base.h File Reference

Classes

- class [__gnu_debug::_Safe_iterator_base](#)
Basic functionality for a safe iterator.
- class [__gnu_debug::_Safe_sequence_base](#)

Base class that supports tracking of iterators that reference a sequence.

Namespaces

- namespace `__gnu_debug`

6.271.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file `safe_base.h`.

6.272 `safe_iterator.h` File Reference

Classes

- class `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>`
Safe iterator wrapper.

Namespaces

- namespace `__gnu_debug`

Functions

- bool `__gnu_debug::__check_singular_aux` (const `_Safe_iterator_base *``__x`)
- template<typename `_IteratorL` , typename `_IteratorR` , typename `_Sequence` >
bool `__gnu_debug::operator!=` (const `_Safe_iterator<_IteratorL, _Sequence>` &`__lhs`, const `_Safe_iterator<_IteratorR, _Sequence>` &`__rhs`)
- template<typename `_Iterator` , typename `_Sequence` >
bool `__gnu_debug::operator!=` (const `_Safe_iterator<_Iterator, _Sequence>` &`__lhs`, const `_Safe_iterator<_Iterator, _Sequence>` &`__rhs`)
- template<typename `_Iterator` , typename `_Sequence` >
`_Safe_iterator<_Iterator, _Sequence>` `__gnu_debug::operator+` (typename `_Safe_iterator<_Iterator, _Sequence>` ::`difference_type` `__n`, const `_Safe_iterator<_Iterator, _Sequence>` &`__i`)
- template<typename `_IteratorL` , typename `_IteratorR` , typename `_Sequence` >
`_Safe_iterator<_IteratorL, _Sequence>` ::`difference_type` `__gnu_debug::operator-` (const `_Safe_iterator<_IteratorL, _Sequence>` &`__lhs`, const `_Safe_iterator<_IteratorR, _Sequence>` &`__rhs`)

- `template<typename _Iterator, typename _Sequence >`
`_Safe_iterator< _Iterator, _Sequence >::difference_type __gnu_`
`debug::operator- (const _Safe_iterator< _Iterator, _Sequence > &__lhs,`
`const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator< (const _Safe_iterator< _Iterator, _Sequence >`
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator< (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator<= (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator<= (const _Safe_iterator< _Iterator, _Sequence`
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _Iterator, _Sequence >`
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence >`
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence`
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`

6.272.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.h](#).

6.273 `safe_iterator.tcc` File Reference

Namespaces

- namespace `__gnu_debug`

6.273.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file `safe_iterator.tcc`.

6.274 `safe_sequence.h` File Reference

Classes

- class `__gnu_debug::_After_nth_from<_Iterator>`
- class `__gnu_debug::_Not_equal_to<_Type>`
- class `__gnu_debug::_Safe_sequence<_Sequence>`

Base class for constructing a safe sequence type that tracks iterators that reference it.

Namespaces

- namespace `__gnu_debug`

6.274.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file `safe_sequence.h`.

6.275 `search.h` File Reference

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace `__gnu_parallel`

Functions

- `template<typename _RAIter, typename _DifferenceTp >`
`void __gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __-`
`length, _DifferenceTp *__off)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >`
`__RAIter1 __gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1`
`__end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`

6.275.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [search.h](#).

6.276 set File Reference

6.276.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [set](#).

6.277 set File Reference

6.277.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set](#).

6.278 set File Reference

6.278.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set](#).

6.279 set.h File Reference

Classes

- class [std::__debug::set<_Key, _Compare, _Allocator>](#)
Class [std::set](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator >
&_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__debug::operator== (const set< _Key, _Compare, _Allocator >
&_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator >
&_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _
Key, _Compare, _Allocator > &__y)`

6.279.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set.h](#).

6.280 set.h File Reference

Classes

- class [std::__profile::set< _Key, _Compare, _Allocator >](#)
Class [std::set](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

6.280.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set.h](#).

6.281 set_operations.h File Reference

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator >
_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b,
std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1,
_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-
Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1,
_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-
Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >
_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1,
_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-
Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter
__begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __-
result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter
__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __-
comp)`

6.281.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [set_operations.h](#).

6.282 settings.h File Reference

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct `__gnu_parallel::_Settings`
class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.

Namespaces

- namespace `__gnu_parallel`

Defines

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

6.282.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

6.282.2 parallelization_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and __off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

Definition in file [settings.h](#).

6.282.3 Define Documentation

6.282.3.1 `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

Parameters

`__c` A condition that is convertible to bool that is overruled by `__gnu_parallel::_Settings::algorithm_strategy`. Usually a decision based on the input size.

Definition at line 95 of file [settings.h](#).

6.283 shared_ptr.h File Reference

Classes

- class [std::enable_shared_from_this<_Tp>](#)
Base class allowing use of member function `shared_from_this`.
- struct [std::hash<shared_ptr<_Tp>>](#)
`std::hash` specialization for `shared_ptr`.

- struct `std::owner_less< shared_ptr< _Tp > >`
Partial specialization of `owner_less` for `shared_ptr`.
- struct `std::owner_less< weak_ptr< _Tp > >`
Partial specialization of `owner_less` for `weak_ptr`.
- class `std::shared_ptr< _Tp >`
A smart pointer with reference-counted copy semantics.
- class `std::weak_ptr< _Tp >`
A smart pointer with weak semantics.

Namespaces

- namespace `std`

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
`shared_ptr< _Tp > std::allocate_shared (_Alloc __a, _Args &&...__args)`
- template<typename _Tp, typename _Tp1 >
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- template<typename _Tp, typename _Tp1 >
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- template<typename _Del, typename _Tp, _Lock_policy _Lp>
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- template<typename _Tp, typename... _Args>
`shared_ptr< _Tp > std::make_shared (_Args &&...__args)`
- template<typename _Tp1, typename _Tp2 >
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- template<typename _Tp1, typename _Tp2 >
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- template<typename _Tp1, typename _Tp2 >
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`

- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

6.283.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared_ptr.h](#).

6.284 `shared_ptr_base.h` File Reference

6.284.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared_ptr_base.h](#).

6.285 `slice_array.h` File Reference

Classes

- class [std::slice](#)
Class defining one-dimensional subset of an array.
- class [std::slice_array< _Tp >](#)
Reference to one-dimensional subset of an array.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

6.285.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [slice_array.h](#).

6.286 slist File Reference

Classes

- class [__gnu_cxx::slist< _Tp, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __gnu_cxx::__slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __gnu_cxx::__slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __node)`
- `_Slist_node_base * __gnu_cxx::__slist_reverse (_Slist_node_base * __node)`
- `size_t __gnu_cxx::__slist_size (_Slist_node_base * __node)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __before_last)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<class _Tp, class _Alloc >
bool __gnu_cxx::operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >
bool __gnu_cxx::operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`

- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator== (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator>= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >`
`void __gnu_cxx::swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`

6.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [slist](#).

6.287 sort.h File Reference

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-Compare __comp, parallel_tag __parallelism)`

- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, multiway_mergesort_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, multiway_mergesort_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, multiway_mergesort_tag __parallelism)`

6.287.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [sort.h](#).

6.288 sso_string_base.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

6.288.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sso_string_base.h](#).

6.289 sstream File Reference

Classes

- class [std::basic_istream< _CharT, _Traits, _Alloc >](#)

Controlling input for `std::string`.

*This class supports reading from objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as *sb*.*

- class [std::basic_ostream< _CharT, _Traits, _Alloc >](#)

Controlling output for `std::string`.

*This class supports writing to objects of type [std::basic_string](#), using the inherited functions from [std::basic_ostream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as *sb*.*

- class [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#)

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic_string](#). (Paraphrased from [27.7.1]/1.).

- class [std::basic_stringstream< _CharT, _Traits, _Alloc >](#)

Controlling input and output for `std::string`.

*This class supports reading from and writing to objects of type [std::basic_string](#), using the inherited functions from [std::basic_iostream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as *sb*.*

Namespaces

- namespace [std](#)

6.289.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [sstream](#).

6.290 `sstream.tcc` File Reference

Namespaces

- namespace [std](#)

6.290.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sstream.tcc](#).

6.291 `stack` File Reference

6.291.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stack](#).

6.292 `standard_policies.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

Enumerations

- enum { `default_store_hash` }

6.292.1 Detailed Description

Contains standard policies for containers.

Definition in file [standard_policies.hpp](#).

6.293 stdatomic.h File Reference

6.293.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdatomic.h](#).

6.294 stdexcept File Reference

Classes

- class [std::domain_error](#)
- class [std::invalid_argument](#)
- class [std::length_error](#)
- class [std::logic_error](#)

One of two subclasses of exception.

- class [std::out_of_range](#)
- class [std::overflow_error](#)
- class [std::range_error](#)
- class [std::runtime_error](#)

One of two subclasses of exception.

- class [std::underflow_error](#)

Namespaces

- namespace [std](#)

6.294.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdexcept](#).

6.295 stdio_filebuf.h File Reference

Classes

- class [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#)

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE's and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

Namespaces

- namespace [__gnu_cxx](#)

6.295.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_filebuf.h](#).

6.296 stdio_sync_filebuf.h File Reference

Classes

- class [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#)

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE's. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

Namespaces

- namespace [__gnu_cxx](#)

6.296.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_sync_filebuf.h](#).

6.297 `std_algo.h` File Reference

Namespaces

- namespace `std`

Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }

Functions

- `template<typename _RandomAccessIterator, typename _Distance >`
`void std::__chunk_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Distance __chunk_size)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`void std::__chunk_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::__copy_n (_InputIterator __first, _Size __n, _-`
`OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::__copy_n (_RandomAccessIterator __first, _Size __n, _-`
`OutputIterator __result, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__final_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__final_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Tp >`
`_RandomAccessIterator std::__find (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::__find (_InputIterator __first, _InputIterator __last, const _Tp`
`&__val, input_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BinaryPredicate >`
`_BidirectionalIterator1 std::__find_end (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_-`
`iterator_tag, _BinaryPredicate __comp)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _-`
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`
`__last2, forward_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _-`
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`
`__last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_-`
`iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator std::find_if (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator std::find_if_not (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred, input_iterator_tag)`
- `template<typename _EuclideanRingElement >`
`_EuclideanRingElement std::gcd (_EuclideanRingElement __m, _-`
`EuclideanRingElement __n)`
- `template<typename _RandomAccessIterator >`
`void std::heap_select (_RandomAccessIterator __first, _-`
`RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::heap_select (_RandomAccessIterator __first, _-`
`RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare`
`__comp)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`
`_ForwardIterator std::inplace_stable_partition (_ForwardIterator __first, _-`
`ForwardIterator __last, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`
`void std::inplace_stable_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::inplace_stable_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __-`
`depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::__introsort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __-`
`depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsort_loop (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::__introsort_loop (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`
`void std::__merge_adaptive (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename`
`_Compare >`
`void std::__merge_adaptive (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1,`
`_Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare`
`__comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BidirectionalIterator3 >`
`_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BidirectionalIterator3, typename _Compare >`
`_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __-`
`comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`
`_Distance >`
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-`
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`
`__step_size)`

- `template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance , typename _Compare >`
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-`
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`
`__step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator , typename _Pointer >`
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >`
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator , typename _Distance >`
`void std::__merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2)`
- `template<typename _BidirectionalIterator , typename _Distance , typename _Compare >`
`void std::__merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator , typename _Compare >`
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _-`
`Compare __comp)`
- `template<typename _Iterator >`
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator , typename _Predicate >`
`_BidirectionalIterator std::__partition (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator _-`
`middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator _-`
`middle, _RandomAccessIterator __last, random_access_iterator_tag)`

- `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance >`
`_BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _-`
`BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __-`
`len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_-`
`size)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Integer __count, const _Tp &__val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp >`
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _-`
`RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_-`
`access_iterator_tag)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _-`
`BinaryPredicate >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`
`std::forward_iterator_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _-`
`BinaryPredicate >`
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _-`
`RandomAccessIter __last, _Integer __count, const _Tp &__val, _-`
`BinaryPredicate __binary_pred, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _-`
`Distance >`
`_ForwardIterator std::__stable_partition_adaptive (_ForwardIterator __first, _-`
`ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer,`
`_Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance , typename`
`_Compare >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,`
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__unguarded_linear_insert (_RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`
`_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator _first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`
`_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator _first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`

- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function`
`__f)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator`
`__gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator _-`
`__middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator _-`
`__middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`
`__pred)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare`
`__comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`

- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __-`
`last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp`
`&__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _`
`Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __-`
`last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _-`
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __-`
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`

```

_FowardIterator1 std::search (_FowardIterator1 __first1, _FowardIterator1 __-
_last1, _FowardIterator2 __first2, _FowardIterator2 __last2, _BinaryPredicate
__predicate)
• template<typename _FowardIterator1, typename _FowardIterator2 >
_FowardIterator1 std::search (_FowardIterator1 __first1, _FowardIterator1 __-
_last1, _FowardIterator2 __first2, _FowardIterator2 __last2)
• template<typename _FowardIterator, typename _Integer, typename _Tp, typename _-
BinaryPredicate >
_FowardIterator std::search\_n (_FowardIterator __first, _FowardIterator __-
last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
• template<typename _FowardIterator, typename _Integer, typename _Tp >
_FowardIterator std::search\_n (_FowardIterator __first, _FowardIterator __-
last, _Integer __count, const _Tp &__val)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-
name _Compare >
_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-
last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,
_Compare __comp)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >
_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-
last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >
_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1
__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-
result)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-
name _Compare >
_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1
__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-
result, _Compare __comp)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >
_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-
InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-
OutputIterator __result)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-
name _Compare >
_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-
InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-
OutputIterator __result, _Compare __comp)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >
_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,
_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
• template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-
name _Compare >
_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,

```

-
- ```

 _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-
 Compare __comp)

```
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _UniformRandomNumberGenerator &__g)`
  - `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
  - `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
  - `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred)`
  - `template<typename _RandomAccessIterator >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
  - `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
  - `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _UnaryOperation __unary_op)`
  - `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_-`  
`op)`
  - `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`  
`_BinaryPredicate __binary_pred)`
  - `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
  - `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result)`
  - `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred)`
  - `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val)`
  - `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val, _Compare __comp)`
-

### 6.297.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_algo.h](#).

## 6.298 stl\_algobase.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define **\_GLIBCXX\_MOVE3**(Tp, Up, Vp)
- #define **\_GLIBCXX\_MOVE\_BACKWARD3**(Tp, Up, Vp)

### Functions

- template<bool \_IsMove, typename \_II, typename \_OI >  
\_OI **std::\_\_copy\_move\_a** (\_II \_\_first, \_II \_\_last, \_OI \_\_result)
- template<bool \_IsMove, typename \_CharT >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT >::\_\_value, ostreambuf\_iterator< \_CharT, char\_traits< \_CharT > > >::\_\_type **std::\_\_copy\_move\_a2** (\_CharT \*, \_CharT \*, ostreambuf\_iterator< \_CharT, char\_traits< \_CharT > >)
- template<bool \_IsMove, typename \_CharT >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT >::\_\_value, ostreambuf\_iterator< \_CharT, char\_traits< \_CharT > > >::\_\_type **std::\_\_copy\_move\_a2** (const \_CharT \*, const \_CharT \*, ostreambuf\_iterator< \_CharT, char\_traits< \_CharT > >)
- template<bool \_IsMove, typename \_CharT >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT >::\_\_value, \_CharT \* >::\_\_type **std::\_\_copy\_move\_a2** (istreambuf\_iterator< \_CharT, char\_traits< \_CharT > >, istreambuf\_iterator< \_CharT, char\_traits< \_CharT > >, \_CharT \*)
- template<bool \_IsMove, typename \_II, typename \_OI >  
\_OI **std::\_\_copy\_move\_a2** (\_II \_\_first, \_II \_\_last, \_OI \_\_result)
- template<bool \_IsMove, typename \_BI1, typename \_BI2 >  
\_BI2 **std::\_\_copy\_move\_backward\_a** (\_BI1 \_\_first, \_BI1 \_\_last, \_BI2 \_\_result)

- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __-`  
`result)`
- `template<typename _II1, typename _II2 >`  
`bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if<!__is_scalar< _Tp >::__value, void >::__type std::__-`  
`fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::__-`  
`fill_a (_Tp * __first, _Tp * __last, const _Tp &__c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if<!__is_scalar< _Tp >::__value, _OutputIterator >::__-`  
`type std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type std::__-`  
`fill_n_a (_Tp * __first, _Size __n, const _Tp &__c)`
- `template<typename _II1, typename _II2 >`  
`bool std::__lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __-`  
`first2, _II2 __last2)`
- `long long std::__lg (long long __n)`
- `long std::__lg (long __n)`
- `template<typename _Size >`  
`_Size std::__lg (_Size __n)`
- `int std::__lg (int __n)`
- `template<typename _Iterator >`  
`_Miter_base< _Iterator >::iterator_type std::__miter_base (_Iterator __it)`
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >::iterator_type std::__niter_base (_Iterator __it)`
- `template<typename _II, typename _OI >`  
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`  
`bool std::equal (_Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _-`  
`BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`  
`value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter\_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 6.298.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_algobase.h](#).

## 6.299 `std_bvector.h` File Reference

### Classes

- struct `std::hash<::vector< bool, _Alloc > >`  
*`std::hash` specialization for `vector<bool>`.*
- class `std::vector< bool, _Alloc >`  
*A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.*

### Namespaces

- namespace `std`

### Typedefs

- typedef unsigned long `std::_Bit_type`

### Enumerations

- enum { `_S_word_bit` }

### Functions

- void `std::_fill_bvector` (`_Bit_iterator __first`, `_Bit_iterator __last`, `bool __x`)
- void `std::fill` (`_Bit_iterator __first`, `_Bit_iterator __last`, `const bool &__x`)
- `_Bit_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_iterator &__x`)
- `_Bit_const_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_const_iterator &__x`)
- `ptrdiff_t std::operator-` (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)

#### 6.299.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `std_bvector.h`.

## 6.300 `std_construct.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _T1, typename... _Args>`  
`void std::\_Construct (_T1 *__p, _Args &&...__args)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator >`  
`void std::\_Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp >`  
`void std::\_Destroy (_Tp *__pointer)`

#### 6.300.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_construct.h](#).

## 6.301 `std_deque.h` File Reference

### Classes

- class [std::\\_Deque\\_base](#)< \_Tp, \_Alloc >
- struct [std::\\_Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr >

*A deque::iterator.*

- class [std::deque](#)< \_Tp, \_Alloc >

*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*



## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

## Functions

- `size_t std::__deque_buf_size (size_t __size)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp`  
`* > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _`  
`Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp`  
`* > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+ (ptrdiff_t __n, const _`  
`Deque_iterator< _Tp, _Ref, _Ptr > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type std::operator- (const _`  
`Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref,`  
`_Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type std::operator- (const`  
`_Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _`  
`RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

### 6.301.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_deque.h](#).

## 6.301.2 Define Documentation

### 6.301.2.1 #define \_GLIBCXX\_DEQUE\_BUF\_SIZE

This function controls the size of memory nodes.

#### Parameters

*size* The size of an element.

#### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 82 of file `stl_deque.h`.

## 6.302 stl\_function.h File Reference

### Classes

- struct `std::binary_function< _Arg1, _Arg2, _Result >`
- class `std::binary_negate< _Predicate >`  
*One of the negation functors.*
- class `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`  
*One of the adaptors for member /// pointers.*
- class `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`  
*One of the adaptors for member /// pointers.*
- class `std::const_mem_fun_ref_t< _Ret, _Tp >`  
*One of the adaptors for member /// pointers.*
- class `std::const_mem_fun_t< _Ret, _Tp >`  
*One of the adaptors for member /// pointers.*
- struct `std::divides< _Tp >`  
*One of the math functors.*

- struct `std::equal_to<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::greater<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::greater_equal<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::less<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::less_equal<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::logical_and<_Tp>`  
*One of the [Boolean operations functors](#).*
- struct `std::logical_not<_Tp>`  
*One of the [Boolean operations functors](#).*
- struct `std::logical_or<_Tp>`  
*One of the [Boolean operations functors](#).*
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun_ref_t<_Ret, _Tp>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun_t<_Ret, _Tp>`  
*One of the [adaptors for member /// pointers](#).*
- struct `std::minus<_Tp>`  
*One of the [math functors](#).*
- struct `std::modulus<_Tp>`  
*One of the [math functors](#).*

- struct `std::multiplies<_Tp>`  
*One of the [math functors](#).*
- struct `std::negate<_Tp>`  
*One of the [math functors](#).*
- struct `std::not_equal_to<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::plus<_Tp>`  
*One of the [math functors](#).*
- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`  
*One of the [adaptors for function pointers](#).*
- class `std::pointer_to_unary_function<_Arg, _Result>`  
*One of the [adaptors for function pointers](#).*
- struct `std::unary_function<_Arg, _Result>`
- class `std::unary_negate<_Predicate>`  
*One of the [negation functors](#).*

## Namespaces

- namespace `std`

## Functions

- template<typename \_Ret, typename \_Tp>  
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- template<typename \_Ret, typename \_Tp, typename \_Arg>  
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- template<typename \_Ret, typename \_Tp, typename \_Arg>  
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(-_Arg))`
- template<typename \_Ret, typename \_Tp>  
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`
- template<typename \_Predicate>  
`unary_negate<_Predicate> std::not1 (const _Predicate &__pred)`

- `template<typename _Predicate >`  
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > std::ptr\_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr\_fun (_Result(*__x)(_Arg1, _Arg2))`

### 6.302.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_function.h](#).

## 6.303 `std_heap.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void std::\_\_adjust\_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void std::\_\_adjust\_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool std::\_\_is\_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool std::\_\_is\_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`bool std::\_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::\_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __-`  
`n, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __-`  
`n)`
- `template<typename _RandomAccessIterator >`  
`void std::__pop_heap (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__pop_heap (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _RandomAccessIterator __result, _Compare`  
`__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _`  
`Compare >`  
`void std::__push_heap (_RandomAccessIterator __first, _Distance __-`  
`holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void std::__push_heap (_RandomAccessIterator __first, _Distance __-`  
`holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`



- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

### 6.303.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `std_heap.h`.

## 6.304 `std_iterator.h` File Reference

### Classes

- class `std::back_insert_iterator< _Container >`  
*Turns assignment into insertion.*
- class `std::front_insert_iterator< _Container >`  
*Turns assignment into insertion.*
- class `std::insert_iterator< _Container >`  
*Turns assignment into insertion.*
- class `std::move_iterator< _Iterator >`
- class `std::reverse_iterator< _Iterator >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Defines

- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

## Functions

- `template<typename _Container >`  
`back_insert_iterator< _Container > std::back\_inserter (_Container &__x)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > std::front\_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator!= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > __gnu_cxx::operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container >::difference_type __gnu_cxx::operator- (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`auto __gnu_cxx::operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)-> decltype(__lhs.base()-__rhs.base())`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator< (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator< (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator >`  
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`

- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator== (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator >`  
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator== (const __normal_iterator< _IteratorL, _`  
`Container > &__lhs, const __normal_iterator< _IteratorR, _Container >`  
`&__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container`  
`> &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const`  
`reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _`  
`Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__`  
`rhs)`

- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`

### 6.304.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

Definition in file [std\\_iterator.h](#).

## 6.305 `std_iterator_base_funcs.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Distance >`  
`void std::__advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void std::__advance (_BidirectionalIterator &__i, _Distance __n,`  
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void std::__advance (_RandomAccessIterator &__i, _Distance __n, random_`  
`access_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`iterator_traits< _RandomAccessIterator >::difference_type std::__distance (_`  
`RandomAccessIterator __first, _RandomAccessIterator __last, random_access_`  
`iterator_tag)`
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type std::__distance (_`  
`InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _InputIterator, typename _Distance >`  
`void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator _`  
`__first, _InputIterator __last)`

- `template<typename _ForwardIterator >`  
`__gnu_cxx::__enable_if< __is_iterator< _ForwardIterator >::__value, _-`  
`ForwardIterator >::__type std::next (_ForwardIterator __x, typename iterator_`  
`traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`  
`__gnu_cxx::__enable_if< __is_iterator< _BidirectionalIterator >::__value, _`  
`BidirectionalIterator >::__type std::prev (_BidirectionalIterator __x, typename`  
`iterator_traits< _BidirectionalIterator >::difference_type __n=1)`

### 6.305.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility functions, such as [distance\(\)](#) and [advance\(\)](#).

Definition in file [stl\\_iterator\\_base\\_funcs.h](#).

## 6.306 stl\_iterator\_base\_types.h File Reference

### Classes

- struct [std::bidirectional\\_iterator\\_tag](#)  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct [std::forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- struct [std::input\\_iterator\\_tag](#)  
*Marking input iterators.*
- struct [std::iterator< \\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference >](#)  
*Common iterator class.*
- struct [std::iterator\\_traits< \\_Iterator >](#)  
*Traits class for iterators.*
- struct [std::iterator\\_traits< \\_Tp \\* >](#)  
*Partial specialization for pointer types.*
- struct [std::iterator\\_traits< const \\_Tp \\* >](#)  
*Partial specialization for const pointer types.*

- struct [std::output\\_iterator\\_tag](#)  
*Marking output iterators.*
- struct [std::random\\_access\\_iterator\\_tag](#)  
*Random-access iterators support a superset of bidirectional /// iterator operations.*

## Namespaces

- namespace [std](#)

## Functions

- template<typename `_Iter` >  
iterator\_traits< `_Iter` >::iterator\_category [std::\\_\\_iterator\\_category](#) (const `_Iter` &)

### 6.306.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

Definition in file [std\\_iterator\\_base\\_types.h](#).

## 6.307 `std_list.h` File Reference

### Classes

- class [std::\\_List\\_base](#)< `_Tp`, `_Alloc` >  
*See [bits/stl\\_deque.h](#)'s `_Deque_base` for an explanation.*
- struct [std::\\_List\\_const\\_iterator](#)< `_Tp` >  
*A `list::const_iterator`.*
- struct [std::\\_List\\_iterator](#)< `_Tp` >  
*A `list::iterator`.*
- struct [std::\\_List\\_node](#)< `_Tp` >

*An actual node in the list.*

- struct [std::\\_List\\_node\\_base](#)

*Common part of a node in the list.*

- class [std::list< \\_Tp, \\_Alloc >](#)

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Val >`  
`bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val >`  
`bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`



### 6.307.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_list.h](#).

## 6.308 `std_map.h` File Reference

### Classes

- class [std::map<\\_Key, \\_Tp, \\_Compare, \\_Alloc >](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const map<_Key, _Tp, _Compare, _Alloc > &__x, const map<_Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const map<_Key, _Tp, _Compare, _Alloc > &__x, const map<_Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const map<_Key, _Tp, _Compare, _Alloc > &__x, const map<_Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const map<_Key, _Tp, _Compare, _Alloc > &__x, const map<_Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator> (const map<_Key, _Tp, _Compare, _Alloc > &__x, const map<_Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const map<_Key, _Tp, _Compare, _Alloc > &__x, const map<_Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap (map<_Key, _Tp, _Compare, _Alloc > &__x, map<_Key, _Tp, _Compare, _Alloc > &__y)`

### 6.308.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_map.h](#).

## 6.309 stl\_multimap.h File Reference

### Classes

- class [std::multimap<\\_Key, \\_Tp, \\_Compare, \\_Alloc>](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,  
const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,  
const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,  
const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,  
const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,  
const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,  
const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>  
void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap<  
_Key, _Tp, _Compare, _Alloc > &__y)`

### 6.309.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_multimap.h](#).

## 6.310 `std_multiset.h` File Reference

### Classes

- class [std::multiset< \\_Key, \\_Compare, \\_Alloc >](#)  
*A standard container made up of elements, which can be retrieved in logarithmic time.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

### 6.310.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_multiset.h](#).

## 6.311 std\_numeric.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Tp >  
_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >  
_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _-  
_BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >  
_OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator _-  
__last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator _-  
__last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _-  
_BinaryOperation1, typename _BinaryOperation2 >  
_Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _-  
_InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _-  
_BinaryOperation2 __binary_op2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >  
_Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _-  
_InputIterator2 __first2, _Tp __init)`
- `template<typename _ForwardIterator, typename _Tp >  
void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _-  
_OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >  
_OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _-  
_OutputIterator __result)`

### 6.311.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_numeric.h](#).

## 6.312 `std_pair.h` File Reference

### Classes

- struct [std::pair<\\_T1, \\_T2>](#)  
*pair holds two objects of arbitrary type.*

### Namespaces

- namespace [std](#)

### Functions

- `template<class _T1, class _T2>`  
`pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair(_T1 &&__x, _T2 &&__y)`
- `template<class _T1, class _T2>`  
`bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`  
`bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`  
`bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`  
`bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`  
`bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`  
`bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`

- `template<class _T1, class _T2 >`  
`void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y)`

### 6.312.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_pair.h](#).

## 6.313 [std\\_queue.h](#) File Reference

### Classes

- class [std::priority\\_queue< \\_Tp, \\_Sequence, \\_Compare >](#)  
*A standard container automatically sorting its contents.*
- class [std::queue< \\_Tp, \\_Sequence >](#)  
*A standard container giving FIFO behavior.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Seq >`  
`bool std::operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`

- `template<typename _Tp, typename _Seq >`  
`bool std::operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`  
`void std::swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y)`
- `template<typename _Tp, typename _Seq >`  
`void std::swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y)`

### 6.313.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_queue.h](#).

## 6.314 `std_raw_storage_iter.h` File Reference

### Classes

- class [std::raw\\_storage\\_iterator< \\_OutputIterator, \\_Tp >](#)

### Namespaces

- namespace [std](#)

### 6.314.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_raw\\_storage\\_iter.h](#).

## 6.315 `std_relops.h` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::rel\\_ops](#)

## Functions

- `template<class _Tp >`  
`bool std::rel\_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator>= (const _Tp &__x, const _Tp &__y)`

### 6.315.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous\\_overloads](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads)

Short summary: the `rel_ops` operators should be avoided for the present.

Definition in file [stl\\_relops.h](#).

## 6.316 `stl_set.h` File Reference

### Classes

- class [std::set< \\_Key, \\_Compare, \\_Alloc >](#)  
*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Key , typename _Compare , typename _Alloc >`



- ```
bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set<
_Key, _Compare, _Alloc > &__y)
```
- ```
template<typename _Key, typename _Compare, typename _Alloc >
bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set<
_Key, _Compare, _Alloc > &__y)
```
  - ```
template<typename _Key, typename _Compare, typename _Alloc >
bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set<
_Key, _Compare, _Alloc > &__y)
```
 - ```
template<typename _Key, typename _Compare, typename _Alloc >
bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set<
_Key, _Compare, _Alloc > &__y)
```
  - ```
template<typename _Key, typename _Compare, typename _Alloc >
bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set<
_Key, _Compare, _Alloc > &__y)
```
 - ```
template<typename _Key, typename _Compare, typename _Alloc >
bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set<
_Key, _Compare, _Alloc > &__y)
```
  - ```
template<typename _Key, typename _Compare, typename _Alloc >
void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare,
_Alloc > &__y)
```

6.316.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_set.h](#).

6.317 `std_stack.h` File Reference

Classes

- class [std::stack< _Tp, _Sequence >](#)
A standard container giving FILO behavior.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Seq >`
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`void std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`

6.317.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_stack.h](#).

6.318 [stl_tempbuf.h](#) File Reference

Classes

- class [std::_Temporary_buffer< _ForwardIterator, _Tp >](#)

Namespaces

- namespace [std](#)

Functions

- `template<typename _ForwardIterator, typename _Tp >`
`void std::__uninitialized_construct_buf` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Tp &__value`)
- `template<typename _Tp >`
`pair<_Tp *, ptrdiff_t > std::get_temporary_buffer` (`ptrdiff_t __len`)
- `template<typename _Tp >`
`void std::return_temporary_buffer` (`_Tp *__p`)

6.318.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_temdbuf.h](#).

6.319 `std_tree.h` File Reference

Namespaces

- namespace [std](#)

Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

Functions

- `std::__attribute__` (`((__pure__))`) `_Rb_tree_node_base *_Rb_tree_increment` (`_Rb_tree_node_base *__x`) throw ()
- `void std::__Rb_tree_insert_and_rebalance` (`const bool __insert_left`, `_Rb_tree_node_base *__x`, `_Rb_tree_node_base *__p`, `_Rb_tree_node_base &__header`) throw ()
- `_Rb_tree_node_base * std::__Rb_tree_rebalance_for_erase` (`_Rb_tree_node_base *const __z`, `_Rb_tree_node_base &__header`) throw ()
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool std::operator!=` (`const _Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x`, `const _Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y`)

- `template<typename _Val >`
`bool std::operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_`
`const_iterator< _Val > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool std::operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool std::operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Val >`
`bool std::operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_`
`tree_const_iterator< _Val > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool std::operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool std::operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool std::operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`void std::swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >`
`&__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `const _Rb_tree_node_base *__root std::throw ()`

6.319.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_tree.h](#).

6.320 `std_uninitialized.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_copy_a (_InputIterator __first, _-`
`InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::__uninitialized_copy_a (_InputIterator __first, _-`
`InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator std::__uninitialized_copy_move (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::__uninitialized_copy_n (_InputIterator __first, _Size _-`
`_n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::__uninitialized_copy_n (_RandomAccessIterator __-`
`first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void std::__uninitialized_default (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void std::__uninitialized_default_a (_ForwardIterator __first, _-`
`ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::__uninitialized_default_a (_ForwardIterator __first, _-`
`ForwardIterator __last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`
`void std::__uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`
`void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _-`
`Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n,`
`allocator< _Tp > &)`

- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _`
`Allocator >`
`_ForwardIterator std::__uninitialized_fill_move (_ForwardIterator __result, _`
`ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator`
`__last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`
`_Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`
`_Tp &__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _`
`InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator std::__uninitialized_move_copy (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _`
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _`
`Allocator >`
`void std::__uninitialized_move_fill (_InputIterator __first1, _InputIterator __-`
`last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _`
`Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _`
`ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &_`
`__x)`

6.320.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_uninitialized.h](#).

6.321 `std_vector.h` File Reference

Classes

- struct [std::_Vector_base< _Tp, _Alloc >](#)
See [bits/stl_deque.h](#)'s [_Deque_base](#) for an explanation.
- class [std::vector< _Tp, _Alloc >](#)
A standard container which offers fixed time access to individual elements in any order.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc >`
`void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`

6.321.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_vector.h](#).

6.322 [stream_iterator.h](#) File Reference

Classes

- class [std::istream_iterator](#)< _Tp, _CharT, _Traits, _Dist >
Provides input iterator semantics for streams.
- class [std::ostream_iterator](#)< _Tp, _CharT, _Traits >
Provides output iterator semantics for streams.

Namespaces

- namespace [std](#)

Functions

- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`

6.322.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stream_iterator.h](#).

6.323 streambuf File Reference

Classes

- class [std::basic_streambuf< _CharT, _Traits >](#)

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits >`
`*, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`
`streamsize std::__copy_streambufs_eof (basic_streambuf< wchar_t > *__-`
`sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)`
- `template<>`
`streamsize std::__copy_streambufs_eof (basic_streambuf< char > *__sbin,`
`basic_streambuf< char > *__sbout, bool &__ineof)`

6.323.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [streambuf](#).

6.324 streambuf.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`streamsize std::__copy_streambufs (basic_streambuf< _CharT, _Traits > *_-`
`_sbin, basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >`
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits >`
`*, basic_streambuf< _CharT, _Traits > *, bool &)`

6.324.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf.tcc](#).

6.325 streambuf_iterator.h File Reference

Classes

- class [std::istreambuf_iterator< _CharT, _Traits >](#)
Provides input iterator semantics for streambufs.
- class [std::ostreambuf_iterator< _CharT, _Traits >](#)
Provides output iterator semantics for streambufs.

Namespaces

- namespace [std](#)

Functions

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::__copy_move_a2 (_CharT *__first, _CharT`
`*__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::__copy_move_a2 (const _CharT *__first,`
`const _CharT *__last, ostreambuf_iterator< _CharT > __result)`

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type`
`std::__copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_`
`iterator< _CharT > __last, _CharT *__result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::copy (istreambuf_iterator< _CharT > _`
`__first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`
`__result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_`
`iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT > _`
`__first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a,`
`const istreambuf_iterator< _CharT, _Traits > &__b)`

6.325.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf_iterator.h](#).

6.326 string File Reference

6.326.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [string](#).

6.327 string File Reference

Classes

- class [__gnu_debug::basic_string< _CharT, _Traits, _Allocator >](#)
Class [std::basic_string](#) with safety/checking/debug instrumentation.

Namespaces

- namespace `__gnu_debug`

Typedefs

- typedef `basic_string< char > __gnu_debug::string`
- typedef `basic_string< wchar_t > __gnu_debug::wstring`

Functions

- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`std::basic_istream< _CharT, _Traits > & __gnu_debug::getline (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`std::basic_istream< _CharT, _Traits > & __gnu_debug::getline (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`bool __gnu_debug::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`bool __gnu_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`bool __gnu_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- template<typename `_CharT`, typename `_Traits`, typename `_Allocator` >
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (`
`_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator< (const _CharT *__lhs, const basic_string< _`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_ostream< _CharT, _Traits > & __gnu_debug::operator<<`
`(std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT,`
`_Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const _CharT *__lhs, const basic_string< _`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic_string< _`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _`
`_Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic_string< _`
`_CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _-`
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_istream< _CharT, _Traits > & __gnu_debug::operator>>`
`(std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits,`
`_Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__-`
`lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

6.327.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/string](#).

6.328 stringfwd.h File Reference

Namespaces

- namespace [std](#)

Typedefs

- `typedef basic_string< char > std::string`
- `typedef basic_string< char16_t > std::u16string`
- `typedef basic_string< char32_t > std::u32string`
- `typedef basic_string< wchar_t > std::wstring`

6.328.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stringfwd.h](#).

6.329 `system_error` File Reference

Classes

- class [std::error_category](#)
error_category
- struct [std::error_code](#)
error_code
- struct [std::error_condition](#)
error_condition
- struct [std::hash< error_code >](#)
std::hash specialization for error_code.
- struct [std::is_error_code_enum< _Tp >](#)
is_error_code_enum
- struct [std::is_error_condition_enum< _Tp >](#)
is_error_condition_enum
- class [std::system_error](#)
Thrown to indicate error code of underlying system.

Namespaces

- namespace [std](#)

Functions

- `_GLIBCXX_CONST const error_category & std::generic_category () throw ()`

- error_code **std::make_error_code** (errc __e)
- error_condition **std::make_error_condition** (errc __e)
- bool **std::operator!=** (const error_code &__lhs, const error_condition &__rhs)
- bool **std::operator!=** (const error_condition &__lhs, const error_condition &__rhs)
- bool **std::operator!=** (const error_code &__lhs, const error_code &__rhs)
- bool **std::operator!=** (const error_condition &__lhs, const error_code &__rhs)
- bool **std::operator<** (const error_code &__lhs, const error_code &__rhs)
- bool **std::operator<** (const error_condition &__lhs, const error_condition &__rhs)
- template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- bool **std::operator==** (const error_condition &__lhs, const error_condition &__rhs)
- bool **std::operator==** (const error_code &__lhs, const error_condition &__rhs)
- bool **std::operator==** (const error_code &__lhs, const error_code &__rhs)
- bool **std::operator==** (const error_condition &__lhs, const error_code &__rhs)
- _GLIBCXX_CONST const error_category & **std::system_category** () throw ()

Variables

- error_code **std::make_error_code** (errc)
- error_condition **std::make_error_condition** (errc)

6.329.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [system_error](#).

6.330 tag_and_trait.hpp File Reference

Classes

- struct [__gnu_pbds::associative_container_tag](#)
Basic associative-container.
- struct [__gnu_pbds::basic_hash_tag](#)

Basic hash.

- struct [__gnu_pbds::basic_tree_tag](#)
Basic tree.
- struct [__gnu_pbds::binary_heap_tag](#)
Binary-heap (array-based).
- struct [__gnu_pbds::binomial_heap_tag](#)
Binomial-heap.
- struct [__gnu_pbds::cc_hash_tag](#)
Collision-chaining hash.
- struct [__gnu_pbds::container_tag](#)
Base data structure tag.
- struct [__gnu_pbds::container_traits< Cntnr >](#)
container_traits
- struct [__gnu_pbds::gp_hash_tag](#)
General-probing hash.
- struct [__gnu_pbds::list_update_tag](#)
List-update.
- struct [__gnu_pbds::null_mapped_type](#)
A mapped-policy indicating that an associative container is a set.
- struct [__gnu_pbds::ov_tree_tag](#)
Ordered-vector tree.
- struct [__gnu_pbds::pairing_heap_tag](#)
Pairing-heap.
- struct [__gnu_pbds::pat_trie_tag](#)
PATRICIA trie.
- struct [__gnu_pbds::priority_queue_tag](#)
Basic priority-queue.
- struct [__gnu_pbds::rb_tree_tag](#)

Red-black tree.

- struct [__gnu_pbds::rc_binomial_heap_tag](#)
Redundant-counter binomial-heap.
- struct [__gnu_pbds::sequence_tag](#)
Basic sequence.
- struct [__gnu_pbds::splay_tree_tag](#)
Splay tree.
- struct [__gnu_pbds::string_tag](#)
Basic string container, inclusive of strings, ropes, etc.
- struct [__gnu_pbds::thin_heap_tag](#)
Thin heap.
- struct [__gnu_pbds::tree_tag](#)
tree.
- struct [__gnu_pbds::trie_tag](#)
trie.

Namespaces

- namespace [__gnu_pbds](#)

Typedefs

- typedef void [__gnu_pbds::trivial_iterator_difference_type](#)

6.330.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file [tag_and_trait.hpp](#).

6.331 tags.h File Reference

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::balanced_quicksort_tag](#)
Forces parallel sorting using balanced quicksort at compile time.
- struct [__gnu_parallel::balanced_tag](#)
Recommends parallel execution using dynamic load-balancing at compile time.
- struct [__gnu_parallel::constant_size_blocks_tag](#)
Selects the constant block size variant for `std::find()`.
- struct [__gnu_parallel::default_parallel_tag](#)
Recommends parallel execution using the default parallel algorithm.
- struct [__gnu_parallel::equal_split_tag](#)
Selects the equal splitting variant for `std::find()`.
- struct [__gnu_parallel::exact_tag](#)
Forces parallel merging with exact splitting, at compile time.
- struct [__gnu_parallel::find_tag](#)
Base class for `std::find()` variants.
- struct [__gnu_parallel::growing_blocks_tag](#)
Selects the growing block size variant for `std::find()`.
- struct [__gnu_parallel::multiway_mergesort_exact_tag](#)
Forces parallel sorting using multiway mergesort with exact splitting at compile time.
- struct [__gnu_parallel::multiway_mergesort_sampling_tag](#)
Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.
- struct [__gnu_parallel::multiway_mergesort_tag](#)
Forces parallel sorting using multiway mergesort at compile time.
- struct [__gnu_parallel::omp_loop_static_tag](#)
Recommends parallel execution using OpenMP static load-balancing at compile time.
- struct [__gnu_parallel::omp_loop_tag](#)
Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

- struct [__gnu_parallel::parallel_tag](#)
Recommends parallel execution at compile time, optionally using a user-specified number of threads.
- struct [__gnu_parallel::quicksort_tag](#)
Forces parallel sorting using unbalanced quicksort at compile time.
- struct [__gnu_parallel::sampling_tag](#)
Forces parallel merging with exact splitting, at compile time.
- struct [__gnu_parallel::sequential_tag](#)
Forces sequential execution at compile time.
- struct [__gnu_parallel::unbalanced_tag](#)
Recommends parallel execution using static load-balancing at compile time.

Namespaces

- namespace [__gnu_parallel](#)

6.331.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [tags.h](#).

6.332 tgmath.h File Reference

6.332.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tgmath.h](#).

6.333 thread File Reference

Classes

- struct [std::hash< thread::id >](#)

[std::hash](#) specialization for [thread::id](#).

- class [std::thread](#)
thread
- class [std::thread::id](#)
thread::id

Namespaces

- namespace [std](#)
- namespace [std::this_thread](#)

Functions

- [thread::id std::this_thread::get_id \(\)](#)
- bool **std::operator!=** ([thread::id __x](#), [thread::id __y](#))
- [template<class _CharT, class _Traits > basic_ostream< _CharT, _Traits > & std::operator<< \(basic_ostream< _CharT, _Traits > &__out, \[thread::id __id\]\(#\)\)](#)
- bool **std::operator<=** ([thread::id __x](#), [thread::id __y](#))
- bool **std::operator>** ([thread::id __x](#), [thread::id __y](#))
- bool **std::operator>=** ([thread::id __x](#), [thread::id __y](#))
- [template<typename _Rep, typename _Period > void std::this_thread::sleep_for \(const chrono::duration< _Rep, _Period > &__rtime\)](#)
- [template<typename _Clock, typename _Duration > void std::this_thread::sleep_until \(const chrono::time_point< _Clock, _Duration > &__atime\)](#)
- void **std::swap** ([thread &__x](#), [thread &__y](#))
- void [std::this_thread::yield \(\)](#)

6.333.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [thread](#).

6.334 `throw_allocator.h` File Reference

Classes

- struct [__gnu_cxx::annotate_base](#)
Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*
- struct [__gnu_cxx::condition_base](#)
Base struct for condition policy.
- struct [__gnu_cxx::forced_error](#)
Thrown by exception safety machinery.
- struct [__gnu_cxx::limit_condition](#)
Base class for incremental control and throw.
- struct [__gnu_cxx::limit_condition::always_adjustor](#)
Always enter the condition.
- struct [__gnu_cxx::limit_condition::limit_adjustor](#)
Enter the nth condition.
- struct [__gnu_cxx::limit_condition::never_adjustor](#)
Never enter the condition.
- struct [__gnu_cxx::random_condition](#)
Base class for random probability control and throw.
- struct [__gnu_cxx::random_condition::always_adjustor](#)
Always enter the condition.
- struct [__gnu_cxx::random_condition::group_adjustor](#)
Group condition.
- struct [__gnu_cxx::random_condition::never_adjustor](#)
Never enter the condition.
- class [__gnu_cxx::throw_allocator_base<_Tp, _Cond>](#)
*Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.
Note: Deallocate not allowed to throw.*

- struct `__gnu_cxx::throw_allocator_limit< _Tp >`
Allocator throwing via limit condition.
- struct `__gnu_cxx::throw_allocator_random< _Tp >`
Allocator throwing via random condition.
- struct `__gnu_cxx::throw_value_base< _Cond >`
Class with exception generation control. Intended to be used as a value_type in templated code.
- struct `__gnu_cxx::throw_value_limit`
Type throwing via limit condition.
- struct `__gnu_cxx::throw_value_random`
Type throwing via random condition.
- struct `std::hash< __gnu_cxx::throw_value_limit >`
Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.
- struct `std::hash< __gnu_cxx::throw_value_random >`
Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Namespaces

- namespace `__gnu_cxx`
- namespace `std`

Functions

- void `__gnu_cxx::__throw_forced_error()`
- template<typename `_Tp`, typename `_Cond`>
bool `__gnu_cxx::operator!=` (const `throw_allocator_base< _Tp, _Cond >` &, const `throw_allocator_base< _Tp, _Cond >` &)
- template<typename `_Cond`>
`throw_value_base< _Cond >` `__gnu_cxx::operator*` (const `throw_value_base< _Cond >` &`__a`, const `throw_value_base< _Cond >` &`__b`)
- template<typename `_Cond`>
`throw_value_base< _Cond >` `__gnu_cxx::operator+` (const `throw_value_base< _Cond >` &`__a`, const `throw_value_base< _Cond >` &`__b`)

- `template<typename _Cond >`
`throw_value_base< _Cond > __gnu_cxx::operator-` (`const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b`)
- `template<typename _Cond >`
`bool __gnu_cxx::operator<` (`const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b`)
- `std::ostream & __gnu_cxx::operator<<` (`std::ostream &os, const annotate_base &__b`)
- `template<typename _Tp, typename _Cond >`
`bool __gnu_cxx::operator==` (`const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &`)
- `template<typename _Cond >`
`bool __gnu_cxx::operator==` (`const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b`)
- `template<typename _Cond >`
`void __gnu_cxx::swap` (`throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b`)

6.334.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

Definition in file [throw_allocator.h](#).

6.335 time_members.h File Reference

Namespaces

- namespace [std](#)

6.335.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [time_members.h](#).

6.336 tree_policy.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- #define `PB_DS_BASE_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`

6.336.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_policy.hpp](#).

6.337 tree_trace_base.hpp File Reference

6.337.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_trace_base.hpp](#).

6.338 trie_policy.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- #define `PB_DS_BASE_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`
- #define `PB_DS_CLASS_T_DEC`

6.338.1 Detailed Description

Contains trie-related policies.

Definition in file [trie_policy.hpp](#).

6.339 tuple File Reference

Classes

- struct [std::_Build_index_tuple<_Num>](#)
Builds an [_Index_tuple](#)<0, 1, 2, ..., _Num-1>.
- struct [std::_Index_tuple<_Indexes>](#)
- struct [std::_Tuple_impl<_Idx>](#)
- struct [std::_Tuple_impl<_Idx, _Head, _Tail...>](#)
- class [std::tuple<_Elements>](#)
tuple
- class [std::tuple<_T1, _T2>](#)
tuple (2-element), with construction and assignment from a pair.
- struct [std::tuple_element<0, tuple<_Head, _Tail...>>](#)
- struct [std::tuple_element<__i, tuple<_Head, _Tail...>>](#)
- struct [std::tuple_size<tuple<_Elements...>>](#)
class tuple_size

Namespaces

- namespace [std](#)

Functions

- template<std::size_t __i, typename _Head, typename... _Tail>
[__add_ref<_Head>::type](#) **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail...> &__t)
- template<std::size_t __i, typename _Head, typename... _Tail>
[__add_c_ref<_Head>::type](#) **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail...> &__t)

- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::tuple_cat_helper (tuple< _TElements...> &&__t, const __index_holder< _TIdx...> &, tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::tuple_cat_helper (const tuple< _TElements...> &__t, const __index_holder< _TIdx...> &, const tuple< _UElements...> &__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::tuple_cat_helper (tuple< _TElements...> &&__t, const __index_holder< _TIdx...> &, const tuple< _UElements...> &__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::tuple_cat_helper (const tuple< _TElements...> &__t, const __index_holder< _TIdx...> &, tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<std::size_t __i, typename... _Elements>`
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type std::get (const tuple< _Elements...> &__t)`
- `template<std::size_t __i, typename... _Elements>`
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type std::get (tuple< _Elements...> &__t)`
- `template<typename... _Elements>`
`tuple< typename __decay_and_strip< _Elements >::__type...> std::make_tuple (_Elements &&...__args)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator!= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator== (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`

- `template<typename... _TElements, typename... _UElements>`
`bool std::operator>= (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `template<typename... _Elements>`
`tuple< typename __pa_add_rvalue_reference< _Elements >::__type...>`
`std::pack_arguments (_Elements &&...__args)`
- `template<typename... _Elements>`
`void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
- `template<typename... _Elements>`
`tuple< _Elements &...> std::tie (_Elements &...__args)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`
`TElements...> &__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`
`TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>`
`&&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>`
`&&__t, const tuple< _UElements...> &__u)`

Variables

- `const _Swallow_assign std::ignore`

6.339.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tuple](#).

6.340 type_traits File Reference

Classes

- struct [std::__declval_protector< _Tp >](#)
declval
- struct [std::add_lvalue_reference< _Tp >](#)
add_lvalue_reference

- struct `std::add_rvalue_reference< _Tp >`
add_rvalue_reference
- struct `std::aligned_storage< _Len, _Align >`
Alignment type.
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
conditional
- class `std::decay< _Tp >`
decay
- struct `std::enable_if< bool, _Tp >`
enable_if
- struct `std::has_nothrow_copy_assign< _Tp >`
has_nothrow_copy_assign
- struct `std::has_nothrow_copy_constructor< _Tp >`
has_nothrow_copy_constructor
- struct `std::has_nothrow_default_constructor< _Tp >`
has_nothrow_default_constructor
- struct `std::has_trivial_copy_assign< _Tp >`
has_trivial_copy_assign
- struct `std::has_trivial_copy_constructor< _Tp >`
has_trivial_copy_constructor
- struct `std::has_trivial_default_constructor< _Tp >`
has_trivial_default_constructor
- struct `std::has_trivial_destructor< _Tp >`
has_trivial_destructor
- struct `std::is_base_of< _Base, _Derived >`
is_base_of
- struct `std::is_constructible< _Tp, _Args >`
is_constructible

- struct `std::is_convertible< _From, _To >`
is_convertible
- struct `std::is_explicitly_convertible< _From, _To >`
is_explicitly_convertible
- struct `std::is_lvalue_reference< typename >`
is_lvalue_reference
- struct `std::is_nothrow_constructible< _Tp, _Args >`
is_nothrow_constructible
- struct `std::is_pod< _Tp >`
is_pod
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference
- struct `std::is_signed< _Tp >`
is_signed
- struct `std::is_standard_layout< _Tp >`
is_standard_layout
- struct `std::is_trivial< _Tp >`
is_trivial
- struct `std::is_unsigned< _Tp >`
is_unsigned
- struct `std::make_signed< _Tp >`
make_signed
- struct `std::make_unsigned< _Tp >`
make_unsigned
- struct `std::remove_reference< _Tp >`
remove_reference

Namespaces

- namespace [std](#)

Functions

- [template<typename _Tp >
add_rvalue_reference< _Tp >::type std::declval \(\) noexcept](#)

6.340.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [type_traits](#).

6.341 type_traits File Reference

Classes

- [struct std::tr1::__is_member_pointer_helper< _Tp >
is_member_pointer](#)
- [struct std::tr1::add_const< _Tp >
add_const](#)
- [struct std::tr1::add_cv< _Tp >
add_cv](#)
- [struct std::tr1::add_pointer< _Tp >
add_pointer](#)
- [struct std::tr1::add_volatile< _Tp >
add_volatile](#)
- [struct std::tr1::alignment_of< _Tp >
alignment_of](#)
- [struct std::tr1::extent< typename, _UInt >
extent](#)
- [struct std::tr1::has_virtual_destructor< _Tp >](#)

has_virtual_destructor

- struct `std::tr1::integral_constant< _Tp, __v >`
integral_constant
- struct `std::tr1::is_abstract< _Tp >`
is_abstract
- struct `std::tr1::is_arithmetic< _Tp >`
is_arithmetic
- struct `std::tr1::is_array< typename >`
is_array
- struct `std::tr1::is_class< _Tp >`
is_class
- struct `std::tr1::is_compound< _Tp >`
is_compound
- struct `std::tr1::is_const< typename >`
is_const
- struct `std::tr1::is_empty< _Tp >`
is_empty
- struct `std::tr1::is_enum< _Tp >`
is_enum
- struct `std::tr1::is_floating_point< _Tp >`
is_floating_point
- struct `std::tr1::is_function< typename >`
is_function
- struct `std::tr1::is_fundamental< _Tp >`
is_fundamental
- struct `std::tr1::is_integral< _Tp >`
is_integral
- struct `std::tr1::is_member_function_pointer< _Tp >`

is_member_function_pointer

- struct `std::tr1::is_member_object_pointer< _Tp >`
is_member_object_pointer
- struct `std::tr1::is_object< _Tp >`
is_object
- struct `std::tr1::is_pointer< _Tp >`
is_pointer
- struct `std::tr1::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::tr1::is_same< typename, typename >`
is_same
- struct `std::tr1::is_scalar< _Tp >`
is_scalar
- struct `std::tr1::is_union< _Tp >`
is_union
- struct `std::tr1::is_void< _Tp >`
is_void
- struct `std::tr1::is_volatile< typename >`
is_volatile
- struct `std::tr1::rank< typename >`
rank
- struct `std::tr1::remove_all_extents< _Tp >`
remove_all_extents
- struct `std::tr1::remove_const< _Tp >`
remove_const
- struct `std::tr1::remove_cv< _Tp >`
remove_cv
- struct `std::tr1::remove_extent< _Tp >`

remove_extent

- struct `std::tr1::remove_pointer<_Tp>`
remove_pointer
- struct `std::tr1::remove_volatile<_Tp>`
remove_volatile

Namespaces

- namespace `std`
- namespace `std::tr1`

Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`

Typedefs

- `typedef integral_constant< bool, false > std::tr1::false_type`
- `typedef integral_constant< bool, true > std::tr1::true_type`

6.341.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/type_traits](#).

6.342 type_traits.h File Reference

Namespaces

- namespace `__gnu_cxx`

Functions

- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type * __ptr)`
- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type)`

6.342.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [type_traits.h](#).

6.343 type_utils.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

Typedefs

- `typedef std::tr1::integral_constant< int, 0 > __gnu_pbds::detail::false_type`
- `typedef std::tr1::integral_constant< int, 1 > __gnu_pbds::detail::true_type`

6.343.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

Definition in file [type_utils.hpp](#).

6.344 typeinfo File Reference

Classes

- class [std::bad_cast](#)

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

- class `std::bad_typeid`

Thrown when a `NULL` pointer in a `typeid` expression is used.

- class `std::type_info`

Part of RTTI.

Namespaces

- namespace `std`

Defines

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`

6.344.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeinfo](#).

6.345 `typelist.h` File Reference

Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_cxx::typelist`

Defines

- `#define _GLIBCXX_TPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`

- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

Functions

- `template<typename Fn, typename Typelist >`
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`

6.345.1 Detailed Description

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file [typelist.h](#).

6.346 types.h File Reference

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Typedefs

- typedef int64_t [__gnu_parallel::_CASable](#)
- typedef uint64_t [__gnu_parallel::_SequenceIndex](#)
- typedef uint16_t [__gnu_parallel::_ThreadIndex](#)

Enumerations

- enum [__gnu_parallel::_AlgorithmStrategy](#) { **heuristic**, **force_sequential**, **force_parallel** }
- enum [__gnu_parallel::_FindAlgorithm](#) { **GROWING_BLOCKS**, **CONSTANT_SIZE_BLOCKS**, **EQUAL_SPLIT** }
- enum [__gnu_parallel::_MultiwayMergeAlgorithm](#) { **LOSER_TREE** }
- enum [__gnu_parallel::_Parallelism](#) {
[__gnu_parallel::sequential](#), [__gnu_parallel::parallel_unbalanced](#), [__gnu_parallel::parallel_balanced](#), [__gnu_parallel::parallel_omp_loop](#),
[__gnu_parallel::parallel_omp_loop_static](#), [__gnu_parallel::parallel_taskqueue](#) }
- enum [__gnu_parallel::_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [__gnu_parallel::_SortAlgorithm](#) { **MWMS**, **QS**, **QS_BALANCED** }
- enum [__gnu_parallel::_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

Variables

- static const int [__gnu_parallel::_CASable_bits](#)
- static const [_CASable](#) [__gnu_parallel::_CASable_mask](#)

6.346.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [types.h](#).

6.347 types_traits.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.347.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file [types_traits.hpp](#).

6.348 unique_copy.h File Reference

Parallel implementations of std::unique_copy(). This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >
_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator >
_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`

6.348.1 Detailed Description

Parallel implementations of std::unique_copy(). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [unique_copy.h](#).

6.349 `unique_ptr.h` File Reference

Classes

- struct `std::default_delete< _Tp >`
Primary template, `default_delete`.
- struct `std::default_delete< _Tp[] >`
Specialization, `default_delete`.
- struct `std::hash< unique_ptr< _Tp, _Tp_Deleter > >`
`std::hash` specialization for `unique_ptr`.
- class `std::unique_ptr< _Tp, _Tp_Deleter >`
20.7.12.2 `unique_ptr` for single objects.
- class `std::unique_ptr< _Tp[], _Tp_Deleter >`
20.7.12.3 `unique_ptr` for array objects with a runtime length

Namespaces

- namespace `std`

Functions

- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator!= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator< (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator<= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator== (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`
`bool std::operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`

- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >
bool std::operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const
unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter >
void std::swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _-
Tp_Deleter > &__y)`

6.349.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unique_ptr.h](#).

6.350 unordered_map File Reference

6.350.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_map](#).

6.351 unordered_map File Reference

Classes

- class [std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_map](#) with safety/checking/debug instrumentation.
- class [std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_multimap](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
>
bool std::__debug::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
>
bool std::__debug::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
>
bool std::__debug::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
>
bool std::__debug::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
>
void std::__debug::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
>
void std::__debug::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

6.351.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_map](#).

6.352 unordered_map File Reference

Classes

- `class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`

Class [`std::unordered_map`](#) wrapper with performance instrumentation.

- class [`std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`](#)

Class [`std::unordered_multimap`](#) wrapper with performance instrumentation.

Namespaces

- namespace [`std`](#)
- namespace [`std::__profile`](#)

Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`void std::__profile::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`void std::__profile::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

6.352.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_map](#).

6.353 unordered_map.h File Reference

Classes

- class [std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class [std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Namespaces

- namespace [std](#)

Functions

- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc , bool __cache_hash_code>`
`bool std::operator!= (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator!= (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator== (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void std::swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void std::swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

6.353.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unordered_map.h](#).

6.354 unordered_set File Reference

6.354.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_set](#).

6.355 unordered_set File Reference

Classes

- class [std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_multiset](#) with safety/checking/debug instrumentation.
- class [std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_set](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__debug::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__debug::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__debug::operator==(const unordered_multiset< _Value, _Hash, _-`
`Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc`
`> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__debug::operator==(const unordered_set< _Value, _Hash, _Pred,`
`_Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__debug::swap(unordered_multiset< _Value, _Hash, _Pred, _Alloc`
`> &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__debug::swap(unordered_set< _Value, _Hash, _Pred, _Alloc > &__-`
`__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

6.355.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_set](#).

6.356 unordered_set File Reference

Classes

- class [std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
Unordered_multiset wrapper with performance instrumentation.
- class [std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >](#)
Unordered_set wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__profile::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__profile::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

6.356.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_set](#).

6.357 unordered_set.h File Reference

Classes

- class [std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class [std::unordered_set< _Value, _Hash, _Pred, _Alloc >](#)
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Namespaces

- namespace [std](#)

Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
bool std::operator!= (const __unordered_set< _Value, _Hash, _Pred, _Alloc,
__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,
_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >
bool std::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc >
&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
bool std::operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _-
Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-
Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >
bool std::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc
> &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >
bool std::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc >
&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
bool std::operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc,
__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,
_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
bool std::operator== (const __unordered_multiset< _Value, _Hash, _Pred, _-
Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-
Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >
bool std::operator== (const unordered_multiset< _Value, _Hash, _Pred, _-
Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &-
__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
void std::swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __-
cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _-
Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
void std::swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_-
hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __-
cache_hash_code > &__y)`

- `template<class _Value , class _Hash , class _Pred , class _Alloc >`
`void std::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc >`
`void std::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

6.357.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unordered_set.h](#).

6.358 utility File Reference

6.358.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [utility](#).

6.359 utility File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

Functions

- `template<std::size_t _Int, class _Tp1 , class _Tp2 >`
`tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::tr1::get`
`(std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, class _Tp1 , class _Tp2 >`
`const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::tr1::get`
`(const std::pair< _Tp1, _Tp2 > &__in)`

6.359.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1_impl/utility](#).

6.360 valarray File Reference

Classes

- class [std::valarray<_Tp>](#)
Smart array designed to support numeric processing.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `template<typename _Tp>
_Expr<_BinClos<__not_equal_to, _ValArray, _ValArray, _Tp, _Tp>, type-
name __fun<__not_equal_to, _Tp>::result_type> std::operator!= (const
valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>
_Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp>, type-
name __fun<__not_equal_to, _Tp>::result_type> std::operator!= (const
valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>
_Expr<_BinClos<__not_equal_to, _Constant, _ValArray, _Tp, _Tp>, type-
name __fun<__not_equal_to, _Tp>::result_type> std::operator!= (const _
Tp &__t, const valarray<_Tp> &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp`
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`
`const valarray< _Tp > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`
`valarray< _Tp > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _-`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _-`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _-`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`
`const valarray< _Tp > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _-`
`_fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`
`&__t, const valarray< _Tp > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _-`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _-`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _-`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _-`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,`
`const valarray< _Tp > &__v)`

6.360.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [valarray](#).

6.361 valarray_after.h File Reference

Namespaces

- namespace [std](#)

Defines

- #define **_DEFINE_EXPR_BINARY_FUNCTION**(_Fun, _UFun)
- #define **_DEFINE_EXPR_BINARY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_EXPR_UNARY_FUNCTION**(_Name, _UName)
- #define **_DEFINE_EXPR_UNARY_OPERATOR**(_Op, _Name)

Functions

- template<class _Dom >
_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type >
std::abs (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > **std::abs** (const valarray<
_Tp > &__v)
- template<class _Dom >
_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type >
std::acos (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > **std::acos** (const valarray<
_Tp > &__v)
- template<class _Dom >
_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type >
std::asin (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > **std::asin** (const valarray<
_Tp > &__v)
- template<class _Dom >
_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type >
std::atan (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > **std::atan** (const valarray<
_Tp > &__v)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, type-
name _Dom1::value_type > **std::atan2** (const _Expr< _Dom1, typename _-
Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_-
type > &__e2)
- template<class _Dom >
_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _-
Dom::value_type >, typename _Dom::value_type > **std::atan2** (const _Expr<
_Dom, typename _Dom::value_type > &__e, const valarray< typename _-
Dom::value_type > &__v)

- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _-`
`Dom >, typename _Dom::value_type > std::atan2 (const valarray< typename`
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value _`
`type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename _Dom::value_type > std::atan2 (const typename _-`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp >`
`std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp >`
`std::atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp >`
`std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type >`
`std::cos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >`
`std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type >`
`std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray<`
`_Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray<`
`_Tp > &__v)`

- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type >`
`std::log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`
`std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const typename _-`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`
`> &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const valarray<`
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`
`std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, type-`
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value-`
`type >::result_type > std::operator% (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __modulus, typename _Dom1::value_type >::result_type >`

std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom >
_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > **std::operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > **std::operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_and, typename _Dom1::value_type >::result_type > **std::operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __logical_and, typename _Dom1::value_type >::result_type >`
`std::operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &_`
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`
`>::result_type > std::operator&& (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`
`type >::result_type > std::operator&& (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`
`>::result_type > std::operator&& (const valarray< typename _Dom::value_`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`
`type >::result_type > std::operator&& (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __multiplies, typename _Dom1::value_type >::result_type >`
`std::operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`
`type >::result_type > std::operator* (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`
`>::result_type > std::operator* (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`
`type >::result_type > std::operator* (const _Expr< _Dom, typename _`

```

Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_
_v)
• template<class _Dom >
  _Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_
type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type
>::result_type > std::operator* (const valarray< typename _Dom::value_type
> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
• template<class _Dom >
  _Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_
type >, typename __fun< __plus, typename _Dom::value_type >::result_type
> std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e,
const valarray< typename _Dom::value_type > &__v)
• template<class _Dom >
  _Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_
type >, typename __fun< __plus, typename _Dom::value_type >::result_type
> std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v,
const typename _Dom::value_type &__t)
• template<class _Dom >
  _Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _
Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type
> std::operator+ (const valarray< typename _Dom::value_type > &__v, const
_Expr< _Dom, typename _Dom::value_type > &__e)
• template<class _Dom1, class _Dom2 >
  _Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename _
fun< __plus, typename _Dom1::value_type >::result_type > std::operator+
(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
_Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
  _Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _
Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type
> std::operator+ (const typename _Dom::value_type &__t, const _Expr< _
Dom, typename _Dom::value_type > &__v)
• template<class _Dom >
  _Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type,
_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_
type > std::operator- (const typename _Dom::value_type &__t, const _Expr<
_Dom, typename _Dom::value_type > &__v)
• template<class _Dom >
  _Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _
Dom::value_type >, typename __fun< __minus, typename _Dom::value_
type >::result_type > std::operator- (const _Expr< _Dom, typename _
Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_
_v)
• template<class _Dom >
  _Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type,

```

_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_type > **std::operator-** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1::value_type >::result_type > **std::operator-** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __minus, typename _Dom::value_type >::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename _Dom1::value_type >::result_type > **std::operator/** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > **std::operator/** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > **std::operator/** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > **std::operator/** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > **std::operator/** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1::value_type >::result_type > **std::operator<** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__v,`
`const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__e,`
`const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`
`std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &_`
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > std::operator<< (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > std::operator<< (const valarray< typename _Dom::value_`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __less_equal, typename _Dom1::value_type >::result_type >`
`std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &_`
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value`
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value`
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > std::operator<= (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > std::operator<= (const valarray< typename _Dom::value`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value`
`type >::result_type > std::operator== (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value`
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`
`>::result_type > std::operator== (const valarray< typename _Dom::value`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value`
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`
`>::result_type > std::operator== (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __equal_to, typename _Dom1::value_type >::result_type >`

```
std::operator== (const _Expr< _Dom1, typename _Dom1::value_type > &_
_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)
```

- ```
template<class _Dom >
_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _
Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_
type >::result_type > std::operator== (const _Expr< _Dom, typename _
Dom::value_type > &_v, const typename _Dom::value_type &_t)
```
- ```
template<class _Dom >
_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _
Dom::value_type >, typename __fun< __greater, typename _Dom::value_
type >::result_type > std::operator> (const _Expr< _Dom, typename _
Dom::value_type > &_v, const typename _Dom::value_type &_t)
```
- ```
template<class _Dom >
_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _
Dom::value_type >, typename __fun< __greater, typename _Dom::value_
type >::result_type > std::operator> (const _Expr< _Dom, typename _
Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_
_v)
```
- ```
template<class _Dom >
_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type,
_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_
type > std::operator> (const valarray< typename _Dom::value_type > &_v,
const _Expr< _Dom, typename _Dom::value_type > &_e)
```
- ```
template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, type-
name __fun< __greater, typename _Dom1::value_type >::result_type >
std::operator> (const _Expr< _Dom1, typename _Dom1::value_type > &_
_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)
```
- ```
template<class _Dom >
_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type,
_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_
type > std::operator> (const typename _Dom::value_type &_t, const _Expr<
_Dom, typename _Dom::value_type > &_v)
```
- ```
template<class _Dom >
_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename
_Dom::value_type >, typename __fun< __greater_equal, typename _
Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom,
typename _Dom::value_type > &_e, const valarray< typename _Dom::value_
type > &_v)
```
- ```
template<class _Dom >
_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _
Dom::value_type, _Dom >, typename __fun< __greater_equal, typename
_Dom::value_type >::result_type > std::operator>= (const typename _
Dom::value_type &_t, const _Expr< _Dom, typename _Dom::value_type >
&_v)
```

- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _-`
`Dom::value_type >::result_type > std::operator>= (const valarray< typename`
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`
`> &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __greater_equal, typename _Dom1::value_type >::result_type >`
`std::operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__-`
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __greater_equal, typename _-`
`Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom,`
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__-`
`_t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`
`std::operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__-`
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`
`type >::result_type > std::operator>> (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`
`>::result_type > std::operator>> (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`
`type >::result_type > std::operator>> (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`
`_v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`
`>::result_type > std::operator>> (const valarray< typename _Dom::value_`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type`

```
>::result_type > std::operator^ (const valarray< typename _Dom::value_type  
> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
```

- template<class _Dom >
_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename _Dom1::value_type >::result_type > **std::operator**^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename _Dom1::value_type >::result_type > **std::operator**| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`
`>::result_type > std::operator| (const valarray< typename _Dom::value_type`
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`
`>::result_type > std::operator|| (const valarray< typename _Dom::value_type`
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`
`type >::result_type > std::operator|| (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`
`type >::result_type > std::operator|| (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`
`>::result_type > std::operator|| (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __logical_or, typename _Dom1::value_type >::result_type >`
`std::operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > std::pow (const valarray< typename`
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename _Dom::value_type > std::pow (const typename _`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow`
`(const valarray< _Tp > &__v, const _Tp &__t)`

- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`
`> &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name _Dom1::value_type > std::pow (const _Expr< _Dom1, typename _`
`Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_`
`type > &__e2)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type >`
`std::sin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type >`
`std::sinh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray<`
`_Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type >`
`std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type >`
`std::tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`
`std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray<`
`_Tp > &__v)`

6.361.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_after.h](#).

6.362 valarray_array.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

Functions

- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t`
`__s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__-`
`restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _-`
`Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _-`
`Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`
`__b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t`
`> __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const`
`size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s,`
`_Tp *__restrict __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __-`
`b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`
`__b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t`
`__n, size_t __s)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const`
`size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n,`
`size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp`
`*__restrict __o)`
- `template<typename _Tp >`
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`

- `template<typename _Tp >`
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`
`_Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp >`
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __-`
`n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __-`
`n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __-`
`n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, const _-`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __-`
`n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __-`
`s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, const _-`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _-`
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::_Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__divides (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented__minus (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _`
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _-`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s,`
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, const Expr< _-`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`
`__i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

6.362.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_array.h](#).

6.363 valarray_array.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _-`
`Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t`
`__n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t`
`__n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a, size_t __s)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`
`__b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e,`
`size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool >`
`__m, const _Tp &__t)`

6.363.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_array.tcc](#).

6.364 `valarray_before.h` File Reference

Namespaces

- namespace [std](#)

6.364.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray_before.h](#).

6.365 `vector` File Reference

6.365.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [vector](#).

6.366 `vector` File Reference

Classes

- class [std::__debug::vector< _Tp, _Allocator >](#)
Class [std::vector](#) with safety/checking/debug instrumentation.
- struct [std::hash< __debug::vector< bool, _Alloc > >](#)
[std::hash](#) specialization for [vector<bool>](#).

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

6.366.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/vector](#).

6.367 vector File Reference

Classes

- struct [std::hash< __profile::vector< bool, _Alloc > >](#)
[std::hash](#) specialization for `vector<bool>`.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

6.367.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/vector](#).

6.368 vector.tcc File Reference

Namespaces

- namespace [std](#)

6.368.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vector.tcc](#).

6.369 vstring.h File Reference

Classes

- class [__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>](#)
Template class [__versa_string](#).
Data structure managing sequences of characters and character-like objects.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- ```
bool __gnu_cxx::operator!= (const _CharT *__lhs, const __versa_string< _-
_CharT, _Traits, _Alloc, _Base > &__rhs)
```
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _-`  
`Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-`  
`rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _-`  
`Base > &__lhs, const _CharT *__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__-`  
`rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_-`  
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`  
`_CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-`  
`rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_-`  
`CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _-`  
`Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-`  
`rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _-`  
`Base > &__lhs, const _CharT *__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

```

bool __gnu_cxx::operator< (const _CharT *__lhs, const __versa_string< _-
_CharT, _Traits, _Alloc, _Base > &__rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-
_CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base > &__str)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-
_rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, const _CharT *__rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _-
_CharT, _Traits, _Alloc, _Base > &__rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _-
_CharT, _Traits, _Alloc, _Base > &__rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _-
_Base > &__lhs, const _CharT *__rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-
_rhs)

• template<typename _CharT, template< typename, typename, typename > class _Base>
__enable_if< std::__is_char< _CharT >::__value, bool >::__type __gnu_-
cxx::operator== (const __versa_string< _CharT, std::char_traits< _CharT >,
std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT,
std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _-
_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-
_rhs)

• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>

```



- bool [\\_\\_gnu\\_cxx::operator<](#) (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator<](#) (const \_CharT \*\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>](#) (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>=](#) (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>=](#) (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>=](#) (const \_CharT \*\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - basic\_istream< \_CharT, \_Traits > & [std::operator>>](#) (basic\_istream< \_CharT, \_Traits > &\_\_is, [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_str)
- double [\\_\\_gnu\\_cxx::stod](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0)
- float [\\_\\_gnu\\_cxx::stof](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0)
- int [\\_\\_gnu\\_cxx::stoi](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- long [\\_\\_gnu\\_cxx::stol](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- long double [\\_\\_gnu\\_cxx::stold](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0)
- long long [\\_\\_gnu\\_cxx::stoll](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long [\\_\\_gnu\\_cxx::stoul](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long long [\\_\\_gnu\\_cxx::stoull](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx, int \_\_base=10)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - void [\\_\\_gnu\\_cxx::swap](#) (\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- \_\_vstring [\\_\\_gnu\\_cxx::to\\_string](#) (long long \_\_val)
- \_\_vstring [\\_\\_gnu\\_cxx::to\\_string](#) (unsigned \_\_val)
- \_\_vstring [\\_\\_gnu\\_cxx::to\\_string](#) (int \_\_val)

- `__vstring __gnu_cxx::to_string` (long \_\_val)
- `__vstring __gnu_cxx::to_string` (unsigned long \_\_val)
- `__vstring __gnu_cxx::to_string` (float \_\_val)
- `__vstring __gnu_cxx::to_string` (unsigned long long \_\_val)
- `__vstring __gnu_cxx::to_string` (long double \_\_val)
- `__vstring __gnu_cxx::to_string` (double \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long long \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (unsigned \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (long \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (long long \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (long double \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (double \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (float \_\_val)
- `__wvstring __gnu_cxx::to_wstring` (int \_\_val)

### 6.369.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [vstring.h](#).

## 6.370 vstring.tcc File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const`  
`_CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`  
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__`  
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_`  
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base`  
`> &__str)`

### 6.370.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring.tcc](#).

## 6.371 `vstring_fwd.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Typedefs

- `typedef __versa_string< char, std::char\_traits< char >, std::allocator< char >,`  
`__rc_string_base > \_\_gnu\_cxx::\_\_rc\_string`
- `typedef __vstring \_\_gnu\_cxx::\_\_sso\_string`

- `typedef __versa_string< char16_t, std::char\_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > __gnu_cxx::__u16rc_string`
- `typedef __u16vstring __gnu_cxx::__u16sso_string`
- `typedef __versa_string< char16_t > __gnu_cxx::__u16vstring`
- `typedef __versa_string< char32_t, std::char\_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > __gnu_cxx::__u32rc_string`
- `typedef __u32vstring __gnu_cxx::__u32sso_string`
- `typedef __versa_string< char32_t > __gnu_cxx::__u32vstring`
- `typedef __versa_string< char > __gnu_cxx::__vstring`
- `typedef __versa_string< wchar_t, std::char\_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > __gnu_cxx::__wrc_string`
- `typedef __wvstring __gnu_cxx::__wsso_string`
- `typedef __versa_string< wchar_t > __gnu_cxx::__wvstring`

### 6.371.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring\\_fwd.h](#).

## 6.372 vstring\_util.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.372.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring\\_util.h](#).

## 6.373 workstealing.h File Reference

Parallelization of embarrassingly parallel execution by means of work-stealing.

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_Job<\\_DifferenceTp>](#)  
*One \_\_job for a certain thread.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- #define `_GLIBCXX_JOB_VOLATILE`

## Functions

- template<typename \_RAIter, typename \_Op, typename \_Fu, typename \_Red, typename \_Result  
>  
\_Op [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing](#) (\_-  
RAIter \_\_begin, \_RAIter \_\_end, \_Op \_\_op, \_Fu &\_\_f, \_Red \_\_r, \_Result \_\_base,  
\_Result &\_\_output, typename [std::iterator\\_traits](#)<\_RAIter>::difference\_type  
\_\_bound)

### 6.373.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing. Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. Journal of the ACM, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [workstealing.h](#).

# Index

- ~\_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase, 1168
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue, 1202
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 1036
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string, 796
- ~auto\_ptr
  - std::auto\_ptr, 1495
- ~basic\_filebuf
  - std::basic\_filebuf, 1514
- ~basic\_fstream
  - std::basic\_fstream, 1554
- ~basic\_ifstream
  - std::basic\_ifstream, 1617
- ~basic\_ios
  - std::basic\_ios, 1668
- ~basic\_iostream
  - std::basic\_iostream, 1704
- ~basic\_istream
  - std::basic\_istream, 1765
- ~basic\_istreamstream
  - std::basic\_istreamstream, 1819
- ~basic\_ofstream
  - std::basic\_ofstream, 1872
- ~basic\_ostream
  - std::basic\_ostream, 1915
- ~basic\_ostringstream
  - std::basic\_ostringstream, 1960
- ~basic\_regex
  - std::basic\_regex, 1997
- ~basic\_streambuf
  - std::basic\_streambuf, 2009
- ~basic\_string
  - std::basic\_string, 2037
- ~basic\_stringstream
  - std::basic\_stringstream, 2128
- ~collate
  - std::collate, 2259
- ~ctype
  - std::ctype< char >, 2300
  - std::ctype< wchar\_t >, 2315
- ~deque
  - std::deque, 2378
- ~facet
  - std::locale::facet, 2641
- ~forward\_list
  - std::forward\_list, 2440
- ~gslice
  - numeric\_arrays, 90
- ~ios\_base
  - std::ios\_base, 2557
- ~locale
  - std::locale, 2634
- ~match\_results
  - std::match\_results, 2685
- ~messages
  - std::messages, 2702
- ~money\_get
  - std::money\_get, 2715
- ~money\_put
  - std::money\_put, 2720
- ~moneypunct
  - std::moneypunct, 2728
- ~num\_get
  - std::num\_get, 2807
- ~num\_put
  - std::num\_put, 2827
- ~numpunct

- 
- std::numpunct, [2873](#)
  - ~sentry
    - std::basic\_ostream::sentry, [1947](#)
  - ~stdio\_filebuf
    - \_\_gnu\_cxx::stdio\_filebuf, [947](#)
  - ~temporary\_buffer
    - \_\_gnu\_cxx::temporary\_buffer, [1001](#)
  - ~time\_get
    - std::time\_get, [3044](#)
  - ~time\_put
    - std::time\_put, [3066](#)
  - ~type\_info
    - std::type\_info, [3116](#)
  - ~vector
    - std::vector, [3156](#)
  - /mnt/share/src/ Directory Reference, [335](#)
  - /mnt/share/src/gcc.svn-trunk/ Directory Reference, [321](#)
  - /mnt/share/src/gcc.svn-trunk/libstdc++-v3/ Directory Reference, [327](#)
  - /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/ Directory Reference, [317](#)
  - /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/ Directory Reference, [318](#)
  - /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/ Directory Reference, [328](#)
  - \_\_gnu\_parallel
    - parallel\_balanced, [403](#)
    - parallel\_omp\_loop, [403](#)
    - parallel\_omp\_loop\_static, [403](#)
    - parallel\_taskqueue, [403](#)
    - parallel\_unbalanced, [403](#)
    - sequential, [403](#)
  - \_AlgorithmStrategy
    - \_\_gnu\_parallel, [403](#)
  - \_BALLOC\_ALIGN\_BYTES
    - bitmap\_allocator.h, [3246](#)
  - \_BinIndex
    - \_\_gnu\_parallel, [402](#)
  - \_Bit\_scan\_forward
    - \_\_gnu\_cxx, [362](#)
  - \_CASable
    - \_\_gnu\_parallel, [402](#)
  - \_CASable\_bits
    - \_\_gnu\_parallel, [450](#)
  - \_CASable\_mask
    - \_\_gnu\_parallel, [450](#)
  - \_Construct
    - std, [614](#)
  - \_DRandomShufflingGlobalData
    - \_\_gnu\_parallel::\_-DRandomShufflingGlobalData, [1137](#)
  - \_Destroy
    - std, [614](#)
  - \_FindAlgorithm
    - \_\_gnu\_parallel, [403](#)
  - \_Find\_first
    - SGIextensions, [17](#)
  - \_Find\_next
    - SGIextensions, [17](#)
  - \_GLIBCXX\_ASSERTIONS
    - compiletime\_settings.h, [3280](#)
  - \_GLIBCXX\_ATOMIC\_PROPERTY
    - atomics, [207](#)
  - \_GLIBCXX\_BAL\_QUICKSORT
    - features.h, [3328](#)
  - \_GLIBCXX\_CALL
    - compiletime\_settings.h, [3280](#)
  - \_GLIBCXX\_DEBUG\_VERIFY
    - macros.h, [3388](#)
  - \_GLIBCXX\_DEPRECATED\_ATTR
    - binders, [124](#)
  - \_GLIBCXX\_DEQUE\_BUF\_SIZE
    - stl\_deque.h, [3528](#)
  - \_GLIBCXX\_FIND\_CONSTANT\_-SIZE\_BLOCKS
    - features.h, [3328](#)
  - \_GLIBCXX\_FIND\_EQUAL\_SPLIT
    - features.h, [3328](#)
  - \_GLIBCXX\_FIND\_GROWING\_-BLOCKS
    - features.h, [3328](#)
  - \_GLIBCXX\_MERGESORT
    - features.h, [3329](#)
  - \_GLIBCXX\_PARALLEL\_CONDITION
    - settings.h, [3497](#)
  - \_GLIBCXX\_PARALLEL\_LENGTH
    - multiway\_merge.h, [3409](#)
-

- `_GLIBCXX_QUICKSORT`
  - `features.h`, 3329
- `_GLIBCXX_RANDOM_SHUFFLE_-`
  - `CONSIDER_L1`
    - `compiletime_settings.h`, 3280
- `_GLIBCXX_RANDOM_SHUFFLE_-`
  - `CONSIDER_TLB`
    - `compiletime_settings.h`, 3280
- `_GLIBCXX_SCALE_DOWN_FPU`
  - `compiletime_settings.h`, 3281
- `_GLIBCXX_TREE_DYNAMIC_-`
  - `BALANCING`
    - `features.h`, 3329
- `_GLIBCXX_TREE_FULL_COPY`
  - `features.h`, 3329
- `_GLIBCXX_TREE_INITIAL_-`
  - `SPLITTING`
    - `features.h`, 3330
- `_GLIBCXX_VERBOSE_LEVEL`
  - `compiletime_settings.h`, 3281
- `_GLIBCXX_VOLATILE`
  - `partition.h`, 3433
  - `queue.h`, 3453
- `_GuardedIterator`
  - `__gnu_parallel::_GuardedIterator`, 1146
- `_LoserTreeBase`
  - `__gnu_parallel::_LoserTreeBase`, 1168
- `_M_allocate_and_copy`
  - `std::vector`, 3156
- `_M_allocate_single_object`
  - `__gnu_cxx::bitmap_allocator`, 865
- `_M_attach`
  - `__gnu_debug::_Safe_iterator`, 1019
  - `__gnu_debug::_Safe_iterator_base`, 1029
- `_M_attach_single`
  - `__gnu_debug::_Safe_iterator`, 1019
  - `__gnu_debug::_Safe_iterator_base`, 1029
- `_M_attached_to`
  - `__gnu_debug::_Safe_iterator`, 1019
  - `__gnu_debug::_Safe_iterator_base`, 1029
- `_M_begin`
  - `__gnu_parallel::_Piece`, 1189
- `_M_bin_proc`
  - `__gnu_parallel::_-`
    - `DRandomShufflingGlobalData`, 1137
- `_M_bins_begin`
  - `__gnu_parallel::_DRSSorterPU`, 1140
- `_M_buf`
  - `__gnu_cxx::enc_filebuf`, 895
  - `__gnu_cxx::stdio_filebuf`, 968
  - `std::basic_filebuf`, 1534
- `_M_buf_locale`
  - `__gnu_cxx::enc_filebuf`, 895
  - `__gnu_cxx::stdio_filebuf`, 968
  - `__gnu_cxx::stdio_sync_filebuf`, 995
  - `std::basic_filebuf`, 1534
  - `std::basic_streambuf`, 2025
  - `std::basic_stringbuf`, 2110
- `_M_buf_size`
  - `__gnu_cxx::enc_filebuf`, 895
  - `__gnu_cxx::stdio_filebuf`, 968
  - `std::basic_filebuf`, 1534
- `_M_can_compare`
  - `__gnu_debug::_Safe_iterator`, 1019
  - `__gnu_debug::_Safe_iterator_base`, 1029
- `_M_clear`
  - `__gnu_cxx::free_list`, 905
- `_M_comp`
  - `__gnu_parallel::_LoserTree`, 1162
  - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1166
  - `__gnu_parallel::_LoserTreeBase`, 1169
- `_M_const_iterators`
  - `__gnu_debug::_Safe_sequence`, 1034
  - `__gnu_debug::_Safe_sequence_base`, 1038
  - `__gnu_debug::basic_string`, 1090
  - `std::__debug::bitset`, 1327
  - `std::__debug::deque`, 1333
  - `std::__debug::list`, 1338
  - `std::__debug::map`, 1344
  - `std::__debug::multimap`, 1349



- std::\_\_debug::multiset, 1354
- std::\_\_debug::set, 1360
- std::\_\_debug::unordered\_map, 1365
- std::\_\_debug::unordered\_multimap, 1369
- std::\_\_debug::unordered\_multiset, 1374
- std::\_\_debug::unordered\_set, 1379
- std::\_\_debug::vector, 1384
- \_M\_create\_node
  - std::list, 2613
- \_M\_create\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 878
  - \_\_gnu\_cxx::stdio\_filebuf, 948
  - std::basic\_filebuf, 1514
- \_M\_data
  - std::\_List\_node, 1445
- \_M\_deallocate\_single\_object
  - \_\_gnu\_cxx::bitmap\_allocator, 865
- \_M\_dereferenceable
  - \_\_gnu\_debug::\_Safe\_iterator, 1020
- \_M\_destroy\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 878
  - \_\_gnu\_cxx::stdio\_filebuf, 948
  - std::basic\_filebuf, 1514
- \_M\_detach
  - \_\_gnu\_debug::\_Safe\_iterator, 1020
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 1029
- \_M\_detach\_all
  - \_\_gnu\_debug::\_Safe\_sequence, 1033
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 1037
  - \_\_gnu\_debug::basic\_string, 1047
  - std::\_\_debug::bitset, 1326
  - std::\_\_debug::deque, 1331
  - std::\_\_debug::list, 1337
  - std::\_\_debug::map, 1342
  - std::\_\_debug::multimap, 1348
  - std::\_\_debug::multiset, 1353
  - std::\_\_debug::set, 1358
  - std::\_\_debug::unordered\_map, 1363
  - std::\_\_debug::unordered\_multimap, 1368
  - std::\_\_debug::unordered\_multiset, 1372
  - std::\_\_debug::unordered\_set, 1377
  - std::\_\_debug::vector, 1383
- \_M\_dist
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 1137
- \_M\_elements\_leftover
  - \_\_gnu\_parallel::\_QSBThreadLocal, 1198
- \_M\_end
  - \_\_gnu\_parallel::\_Piece, 1189
- \_M\_ext\_buf
  - \_\_gnu\_cxx::enc\_filebuf, 895
  - \_\_gnu\_cxx::stdio\_filebuf, 969
  - std::basic\_filebuf, 1534
- \_M\_ext\_buf\_size
  - \_\_gnu\_cxx::enc\_filebuf, 895
  - \_\_gnu\_cxx::stdio\_filebuf, 969
  - std::basic\_filebuf, 1534
- \_M\_ext\_next

- \_\_gnu\_cxx::enc\_filebuf, 896
- \_\_gnu\_cxx::stdio\_filebuf, 969
- std::basic\_filebuf, 1535
- \_M\_fill\_initialize
  - std::deque, 2378
- \_M\_finish\_iterator
  - \_\_gnu\_parallel::\_\_accumulate\_selector, 1093
  - \_\_gnu\_parallel::\_\_adjacent\_difference\_selector, 1094
  - \_\_gnu\_parallel::\_\_count\_if\_selector, 1101
  - \_\_gnu\_parallel::\_\_count\_selector, 1103
  - \_\_gnu\_parallel::\_\_fill\_selector, 1105
  - \_\_gnu\_parallel::\_\_for\_each\_selector, 1111
  - \_\_gnu\_parallel::\_\_generate\_selector, 1112
  - \_\_gnu\_parallel::\_\_generic\_for\_each\_selector, 1115
  - \_\_gnu\_parallel::\_\_identity\_selector, 1116
  - \_\_gnu\_parallel::\_\_inner\_product\_selector, 1119
  - \_\_gnu\_parallel::\_\_replace\_if\_selector, 1128
  - \_\_gnu\_parallel::\_\_replace\_selector, 1131
  - \_\_gnu\_parallel::\_\_transform1\_selector, 1132
  - \_\_gnu\_parallel::\_\_transform2\_selector, 1134
- \_M\_first
  - \_\_gnu\_parallel::\_Job, 1153
- \_M\_first\_insert
  - \_\_gnu\_parallel::\_LoserTree, 1162
  - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1166
  - \_\_gnu\_parallel::\_LoserTreeBase, 1169
- \_M\_gcount
  - std::basic\_fstream, 1599
  - std::basic\_ifstream, 1652
  - std::basic\_iostream, 1748
  - std::basic\_istream, 1799
  - std::basic\_istreamream, 1854
  - std::basic\_stringstream, 2172
- \_M\_get
  - \_\_gnu\_cxx::free\_list, 905
- \_M\_get\_distance
  - \_\_gnu\_debug::\_Safe\_iterator, 1020
- \_M\_get\_mutex
  - \_\_gnu\_debug::\_Safe\_iterator, 1020
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 1029
  - \_\_gnu\_debug::\_Safe\_sequence, 1033
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 1037
  - \_\_gnu\_debug::basic\_string, 1047
  - std::\_\_debug::bitset, 1326
  - std::\_\_debug::deque, 1331
  - std::\_\_debug::list, 1337
  - std::\_\_debug::map, 1343
  - std::\_\_debug::multimap, 1348
  - std::\_\_debug::multiset, 1353
  - std::\_\_debug::set, 1359
  - std::\_\_debug::unordered\_map, 1363
  - std::\_\_debug::unordered\_multimap, 1368
  - std::\_\_debug::unordered\_multiset, 1373
  - std::\_\_debug::unordered\_set, 1377
  - std::\_\_debug::vector, 1383
- \_M\_get\_result
  - std::\_\_basic\_future, 1303
  - std::future, 2472
  - std::future< \_Res & >, 2475
  - std::future< void >, 2477
  - std::shared\_future, 2999
  - std::shared\_future< \_Res & >, 3002
  - std::shared\_future< void >, 3005
- \_M\_getloc
  - std::basic\_fstream, 1555
  - std::basic\_ifstream, 1618
  - std::basic\_ios, 1669
  - std::basic\_iostream, 1704
  - std::basic\_istream, 1766
  - std::basic\_istreamream, 1820
  - std::basic\_ofstream, 1872
  - std::basic\_ostream, 1915

- 
- std::basic\_ostringstream, 1960
  - std::basic\_stringstream, 2128
  - std::ios\_base, 2557
  - \_M\_global
    - \_\_gnu\_parallel::\_QSBThreadLocal, 1198
  - \_M\_in\_beg
    - \_\_gnu\_cxx::enc\_filebuf, 896
    - \_\_gnu\_cxx::stdio\_filebuf, 969
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 995
  - std::basic\_filebuf, 1535
  - std::basic\_streambuf, 2026
  - std::basic\_stringbuf, 2110
  - \_M\_in\_cur
    - \_\_gnu\_cxx::enc\_filebuf, 896
    - \_\_gnu\_cxx::stdio\_filebuf, 970
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 995
  - std::basic\_filebuf, 1535
  - std::basic\_streambuf, 2026
  - std::basic\_stringbuf, 2110
  - \_M\_in\_end
    - \_\_gnu\_cxx::enc\_filebuf, 896
    - \_\_gnu\_cxx::stdio\_filebuf, 970
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 996
  - std::basic\_filebuf, 1536
  - std::basic\_streambuf, 2026
  - std::basic\_stringbuf, 2110
  - \_M\_incrementable
    - \_\_gnu\_debug::\_Safe\_iterator, 1021
  - \_M\_initial
    - \_\_gnu\_parallel::\_QSBThreadLocal, 1198
  - \_M\_initialize\_map
    - std::\_Deque\_base, 1427
    - std::deque, 2379
  - \_M\_insert
    - \_\_gnu\_cxx::free\_list, 905
  - \_M\_invalidate
    - \_\_gnu\_debug::\_Safe\_iterator, 1021
  - \_M\_invalidate\_all
    - \_\_gnu\_debug::\_Safe\_sequence, 1033
    - \_\_gnu\_debug::\_Safe\_sequence\_base, 1037
    - \_\_gnu\_debug::basic\_string, 1047
    - std::\_\_debug::bitset, 1326
    - std::\_\_debug::deque, 1332
    - std::\_\_debug::list, 1337
    - std::\_\_debug::map, 1343
    - std::\_\_debug::multimap, 1348
    - std::\_\_debug::multiset, 1353
    - std::\_\_debug::set, 1359
    - std::\_\_debug::unordered\_map, 1364
    - std::\_\_debug::unordered\_multimap, 1368
    - std::\_\_debug::unordered\_multiset, 1373
    - std::\_\_debug::unordered\_set, 1378
    - std::\_\_debug::vector, 1383
  - \_M\_invalidate\_if
    - \_\_gnu\_debug::\_Safe\_sequence, 1033
    - \_\_gnu\_debug::basic\_string, 1047
    - std::\_\_debug::deque, 1332
    - std::\_\_debug::list, 1338
    - std::\_\_debug::map, 1343
    - std::\_\_debug::multimap, 1348
    - std::\_\_debug::multiset, 1353
    - std::\_\_debug::set, 1359
    - std::\_\_debug::unordered\_map, 1364
    - std::\_\_debug::unordered\_multimap, 1368
    - std::\_\_debug::unordered\_multiset, 1373
    - std::\_\_debug::unordered\_set, 1378
    - std::\_\_debug::vector, 1384
  - \_M\_invalidate\_single
    - \_\_gnu\_debug::\_Safe\_iterator, 1021
  - \_M\_is\_begin
    - \_\_gnu\_debug::\_Safe\_iterator, 1022
  - \_M\_is\_end
    - \_\_gnu\_debug::\_Safe\_iterator, 1022
  - \_M\_iterators
    - \_\_gnu\_debug::\_Safe\_sequence, 1034
    - \_\_gnu\_debug::\_Safe\_sequence\_base, 1038
    - \_\_gnu\_debug::basic\_string, 1090
    - std::\_\_debug::bitset, 1327
    - std::\_\_debug::deque, 1333
    - std::\_\_debug::list, 1339
    - std::\_\_debug::map, 1344
-

- std::\_\_debug::multimap, [1349](#)
- std::\_\_debug::multiset, [1354](#)
- std::\_\_debug::set, [1360](#)
- std::\_\_debug::unordered\_map, [1365](#)
- std::\_\_debug::unordered\_multimap, [1369](#)
- std::\_\_debug::unordered\_multiset, [1374](#)
- std::\_\_debug::unordered\_set, [1379](#)
- std::\_\_debug::vector, [1385](#)
- \_M\_key
  - \_\_gnu\_parallel::\_LoserTreeBase::\_Loser, [1171](#)
- \_M\_last
  - \_\_gnu\_parallel::\_Job, [1153](#)
- \_M\_leftover\_parts
  - \_\_gnu\_parallel::\_QSBThreadLocal, [1198](#)
- \_M\_load
  - \_\_gnu\_parallel::\_Job, [1153](#)
- \_M\_log\_k
  - \_\_gnu\_parallel::\_LoserTree, [1162](#)
  - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, [1166](#)
  - \_\_gnu\_parallel::\_LoserTreeBase, [1170](#)
- \_M\_losers
  - \_\_gnu\_parallel::\_LoserTree, [1162](#)
  - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, [1166](#)
  - \_\_gnu\_parallel::\_LoserTreeBase, [1170](#)
- \_M\_mode
  - \_\_gnu\_cxx::enc\_filebuf, [897](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [970](#)
  - std::basic\_filebuf, [1536](#)
  - std::basic\_stringbuf, [2111](#)
- \_M\_new\_elements\_at\_back
  - std::deque, [2379](#)
- \_M\_new\_elements\_at\_front
  - std::deque, [2379](#)
- \_M\_next
  - \_\_gnu\_debug::\_Safe\_iterator, [1025](#)
  - \_\_gnu\_debug::\_Safe\_iterator\_base, [1030](#)
- \_M\_num\_bins
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, [1138](#)
- \_M\_num\_bits
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, [1138](#)
- \_M\_num\_threads
  - \_\_gnu\_parallel::\_DRSSorterPU, [1140](#)
  - \_\_gnu\_parallel::\_PMWMSortingData, [1192](#)
  - \_\_gnu\_parallel::\_QSBThreadLocal, [1198](#)
- \_M\_offsets
  - \_\_gnu\_parallel::\_PMWMSortingData, [1192](#)
- \_M\_out\_beg
  - \_\_gnu\_cxx::enc\_filebuf, [897](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [971](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [996](#)
  - std::basic\_filebuf, [1536](#)
  - std::basic\_streambuf, [2027](#)
  - std::basic\_stringbuf, [2111](#)
- \_M\_out\_cur
  - \_\_gnu\_cxx::enc\_filebuf, [897](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [971](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [997](#)
  - std::basic\_filebuf, [1537](#)
  - std::basic\_streambuf, [2027](#)
  - std::basic\_stringbuf, [2111](#)
- \_M\_out\_end
  - \_\_gnu\_cxx::enc\_filebuf, [898](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [971](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [997](#)
  - std::basic\_filebuf, [1537](#)
  - std::basic\_streambuf, [2027](#)
  - std::basic\_stringbuf, [2112](#)
- \_M\_pback
  - \_\_gnu\_cxx::enc\_filebuf, [898](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [972](#)
  - std::basic\_filebuf, [1537](#)
- \_M\_pback\_cur\_save
  - \_\_gnu\_cxx::enc\_filebuf, [898](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [972](#)
  - std::basic\_filebuf, [1537](#)

- 
- `_M_pback_end_save`
    - `__gnu_cxx::enc_filebuf`, 898
    - `__gnu_cxx::stdio_filebuf`, 972
    - `std::basic_filebuf`, 1538
  - `_M_pback_init`
    - `__gnu_cxx::enc_filebuf`, 899
    - `__gnu_cxx::stdio_filebuf`, 972
    - `std::basic_filebuf`, 1538
  - `_M_pieces`
    - `__gnu_parallel::_-`  
PMWMSortingData, 1192
  - `_M_pop_back_aux`
    - `std::deque`, 2380
  - `_M_pop_front_aux`
    - `std::deque`, 2380
  - `_M_prior`
    - `__gnu_debug::_Safe_iterator`, 1025
    - `__gnu_debug::_Safe_iterator_base`, 1030
  - `_M_push_back_aux`
    - `std::deque`, 2380
  - `_M_push_front_aux`
    - `std::deque`, 2380
  - `_M_range_check`
    - `std::deque`, 2381
    - `std::vector`, 3156
  - `_M_range_initialize`
    - `std::deque`, 2381
  - `_M_reading`
    - `__gnu_cxx::enc_filebuf`, 899
    - `__gnu_cxx::stdio_filebuf`, 973
    - `std::basic_filebuf`, 1538
  - `_M_reallocate_map`
    - `std::deque`, 2382
  - `_M_reserve_elements_at_back`
    - `std::deque`, 2382
  - `_M_reserve_elements_at_front`
    - `std::deque`, 2382
  - `_M_reserve_map_at_back`
    - `std::deque`, 2383
  - `_M_reserve_map_at_front`
    - `std::deque`, 2383
  - `_M_revalidate_singular`
    - `__gnu_debug::_Safe_sequence`, 1033
  - `__gnu_debug::_Safe_sequence_-  
base`, 1037
  - `__gnu_debug::basic_string`, 1048
  - `std::__debug::bitset`, 1327
  - `std::__debug::deque`, 1332
  - `std::__debug::list`, 1338
  - `std::__debug::map`, 1343
  - `std::__debug::multimap`, 1348
  - `std::__debug::multiset`, 1354
  - `std::__debug::set`, 1359
  - `std::__debug::unordered_map`, 1364
  - `std::__debug::unordered_multimap`, 1369
  - `std::__debug::unordered_multiset`, 1373
  - `std::__debug::unordered_set`, 1378
  - `std::__debug::vector`, 1384
  - `_M_samples`
    - `__gnu_parallel::_-`  
PMWMSortingData, 1193
  - `_M_sd`
    - `__gnu_parallel::_DRSSorterPU`, 1140
  - `_M_seed`
    - `__gnu_parallel::_DRSSorterPU`, 1141
  - `_M_sequence`
    - `__gnu_debug::_Safe_iterator`, 1025
    - `__gnu_debug::_Safe_iterator_base`, 1030
  - `_M_sequential_algorithm`
    - `__gnu_parallel::__adjacent_find_-  
selector`, 1096
    - `__gnu_parallel::__find_first_of_-  
selector`, 1107
    - `__gnu_parallel::__find_if_selector`, 1108
    - `__gnu_parallel::__mismatch_-  
selector`, 1122
  - `_M_set_buffer`
    - `__gnu_cxx::enc_filebuf`, 878
    - `__gnu_cxx::stdio_filebuf`, 948
    - `std::basic_filebuf`, 1514
  - `_M_set_node`
    - `std::_Deque_iterator`, 1430
  - `_M_singular`
-

- \_\_gnu\_debug::\_Safe\_iterator, 1022
- \_\_gnu\_debug::\_Safe\_iterator\_base, 1030
- \_M\_source
  - \_\_gnu\_parallel::\_-
    - DRandomShufflingGlobalData, 1138
  - \_\_gnu\_parallel::\_LoserTreeBase::\_-
    - Loser, 1171
  - \_\_gnu\_parallel::\_-
    - PMWMSSortingData, 1193
- \_M\_starts
  - \_\_gnu\_parallel::\_-
    - DRandomShufflingGlobalData, 1138
  - \_\_gnu\_parallel::\_-
    - PMWMSSortingData, 1193
- \_M\_sup
  - \_\_gnu\_parallel::\_LoserTreeBase::\_-
    - Loser, 1171
- \_M\_swap
  - \_\_gnu\_debug::\_Safe\_sequence, 1034
  - \_\_gnu\_debug::\_Safe\_sequence\_-
    - base, 1037
  - \_\_gnu\_debug::basic\_string, 1048
  - std::\_\_debug::bitset, 1327
  - std::\_\_debug::deque, 1332
  - std::\_\_debug::list, 1338
  - std::\_\_debug::map, 1343
  - std::\_\_debug::multimap, 1349
  - std::\_\_debug::multiset, 1354
  - std::\_\_debug::set, 1359
  - std::\_\_debug::unordered\_map, 1364
  - std::\_\_debug::unordered\_multimap, 1369
  - std::\_\_debug::unordered\_multiset, 1373
  - std::\_\_debug::unordered\_set, 1378
  - std::\_\_debug::vector, 1384
- \_M\_temporaries
  - \_\_gnu\_parallel::\_-
    - DRandomShufflingGlobalData, 1139
- \_M\_temporary
  - \_\_gnu\_parallel::\_-
    - PMWMSSortingData, 1193
- \_M\_transfer\_iter
  - \_\_gnu\_debug::\_Safe\_sequence, 1034
  - \_\_gnu\_debug::basic\_string, 1048
  - std::\_\_debug::deque, 1332
  - std::\_\_debug::list, 1338
  - std::\_\_debug::map, 1344
  - std::\_\_debug::multimap, 1349
  - std::\_\_debug::multiset, 1354
  - std::\_\_debug::set, 1360
  - std::\_\_debug::unordered\_map, 1364
  - std::\_\_debug::unordered\_multimap, 1369
  - std::\_\_debug::unordered\_multiset, 1373
  - std::\_\_debug::unordered\_set, 1378
  - std::\_\_debug::vector, 1384
- \_M\_use\_pointer
  - \_\_gnu\_parallel::\_LoserTreeTraits, 1181
- \_M\_version
  - \_\_gnu\_debug::\_Safe\_iterator, 1026
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 1031
  - \_\_gnu\_debug::\_Safe\_sequence, 1035
  - \_\_gnu\_debug::\_Safe\_sequence\_-
    - base, 1038
  - \_\_gnu\_debug::basic\_string, 1090
  - std::\_\_debug::bitset, 1327
  - std::\_\_debug::deque, 1333
  - std::\_\_debug::list, 1339
  - std::\_\_debug::map, 1344
  - std::\_\_debug::multimap, 1349
  - std::\_\_debug::multiset, 1355
  - std::\_\_debug::set, 1360
  - std::\_\_debug::unordered\_map, 1365
  - std::\_\_debug::unordered\_multimap, 1370
  - std::\_\_debug::unordered\_multiset, 1374
  - std::\_\_debug::unordered\_set, 1379
  - std::\_\_debug::vector, 1385
- \_M\_w

- 
- std::\_Base\_bitset, 1422
  - \_M\_write
    - std::basic\_fstream, 1555
    - std::basic\_istream, 1704
    - std::basic\_ofstream, 1873
    - std::basic\_ostream, 1916
    - std::basic\_ostringstream, 1961
    - std::basic\_stringstream, 2129
  - \_MultiwayMergeAlgorithm
    - \_\_gnu\_parallel, 403
  - \_Parallelism
    - \_\_gnu\_parallel, 403
  - \_PartialSumAlgorithm
    - \_\_gnu\_parallel, 403
  - \_Piece
    - \_\_gnu\_parallel::\_QSBThreadLocal, 1197
  - \_PseudoSequence
    - \_\_gnu\_parallel::\_PseudoSequence, 1194
  - \_QSBThreadLocal
    - \_\_gnu\_parallel::\_QSBThreadLocal, 1197
  - \_RandomNumber
    - \_\_gnu\_parallel::\_RandomNumber, 1199
  - \_RestrictedBoundedConcurrentQueue
    - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue, 1202
  - \_Safe\_iterator
    - \_\_gnu\_debug::\_Safe\_iterator, 1017, 1018
  - \_Safe\_iterator\_base
    - \_\_gnu\_debug::\_Safe\_iterator\_base, 1028
  - \_SequenceIndex
    - \_\_gnu\_parallel, 402
  - \_SortAlgorithm
    - \_\_gnu\_parallel, 403
  - \_SplittingAlgorithm
    - \_\_gnu\_parallel, 404
  - \_Temporary\_buffer
    - std::\_Temporary\_buffer, 1464
  - \_ThreadIndex
    - \_\_gnu\_parallel, 402
  - \_Unchecked\_flip
    - SGIextensions, 18
  - \_Unchecked\_reset
    - SGIextensions, 18
  - \_Unchecked\_set
    - SGIextensions, 18
  - \_Unchecked\_test
    - SGIextensions, 18
  - \_\_atomic0::atomic\_address, 763
  - \_\_atomic0::atomic\_bool, 764
  - \_\_atomic0::atomic\_flag, 765
  - \_\_atomic2::atomic\_address, 765
  - \_\_atomic2::atomic\_bool, 766
  - \_\_atomic2::atomic\_flag, 767
  - \_\_begin1\_iterator
    - \_\_gnu\_parallel::\_inner\_product\_selector, 1118
  - \_\_begin2\_iterator
    - \_\_gnu\_parallel::\_inner\_product\_selector, 1119
  - \_\_bins\_end
    - \_\_gnu\_parallel::\_DRSSorterPU, 1140
  - \_\_bit\_allocate
    - \_\_gnu\_cxx::\_\_detail, 374
  - \_\_bit\_free
    - \_\_gnu\_cxx::\_\_detail, 374
  - \_\_calc\_borders
    - \_\_gnu\_parallel, 404
  - \_\_check\_dereferenceable
    - \_\_gnu\_debug, 382
  - \_\_check\_singular
    - \_\_gnu\_debug, 383
  - \_\_check\_singular\_aux
    - \_\_gnu\_debug, 383
  - \_\_check\_string
    - \_\_gnu\_debug, 383
  - \_\_compare\_and\_swap
    - \_\_gnu\_parallel, 404
  - \_\_compare\_and\_swap\_32
    - \_\_gnu\_parallel, 405
  - \_\_compare\_and\_swap\_64
    - \_\_gnu\_parallel, 405
  - \_\_ctype\_type
    - std::basic\_fstream, 1547
    - std::basic\_ifstream, 1612
-

- std::basic\_ios, 1663
- std::basic\_iostream, 1697
- std::basic\_istream, 1760
- std::basic\_istream, 1814
- std::basic\_ofstream, 1867
- std::basic\_ostream, 1910
- std::basic\_ostringstream, 1955
- std::basic\_stringstream, 2121
- \_\_cxxabiv1::\_\_forced\_unwind, 767
- \_\_decode2
  - \_\_gnu\_parallel, 405
- \_\_delete\_min\_insert
  - \_\_gnu\_parallel::LoserTree, 1161
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 1164
- \_\_determine\_samples
  - \_\_gnu\_parallel, 406
- \_\_encode2
  - \_\_gnu\_parallel, 406
- \_\_env\_t
  - \_\_gnu\_profile, 459
- \_\_fetch\_and\_add
  - \_\_gnu\_parallel, 407
- \_\_fetch\_and\_add\_32
  - \_\_gnu\_parallel, 407
- \_\_fetch\_and\_add\_64
  - \_\_gnu\_parallel, 408
- \_\_final\_insertion\_sort
  - std, 601
- \_\_find
  - std, 601
- \_\_find\_if
  - std, 601, 602
- \_\_find\_if\_not
  - std, 602
- \_\_find\_template
  - \_\_gnu\_parallel, 408–410
- \_\_for\_each\_template\_random\_access
  - \_\_gnu\_parallel, 411
- \_\_for\_each\_template\_random\_access\_ed
  - \_\_gnu\_parallel, 411
- \_\_for\_each\_template\_random\_access\_-omp\_loop
  - \_\_gnu\_parallel, 412
- \_\_for\_each\_template\_random\_access\_-omp\_loop\_static
  - \_\_gnu\_parallel, 413
- \_\_for\_each\_template\_random\_access\_-workstealing
  - \_\_gnu\_parallel, 413
- \_\_gcd
  - std, 602
- \_\_genrand\_bits
  - \_\_gnu\_parallel::RandomNumber, 1200
- \_\_get\_\_global\_lock
  - \_\_gnu\_profile, 459
- \_\_get\_min\_source
  - \_\_gnu\_parallel::LoserTree, 1161
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 1164
  - \_\_gnu\_parallel::LoserTreeBase, 1169
- \_\_get\_num\_threads
  - \_\_gnu\_parallel::balanced\_-quicksort\_tag, 1216
  - \_\_gnu\_parallel::balanced\_tag, 1218
  - \_\_gnu\_parallel::default\_parallel\_-tag, 1220
  - \_\_gnu\_parallel::exact\_tag, 1223
  - \_\_gnu\_parallel::multiway\_-mergesort\_exact\_tag, 1226
  - \_\_gnu\_parallel::multiway\_-mergesort\_sampling\_tag, 1228
  - \_\_gnu\_parallel::multiway\_-mergesort\_tag, 1229
  - \_\_gnu\_parallel::omp\_loop\_static\_-tag, 1231
  - \_\_gnu\_parallel::omp\_loop\_tag, 1232
  - \_\_gnu\_parallel::parallel\_tag, 1235
  - \_\_gnu\_parallel::quicksort\_tag, 1237
  - \_\_gnu\_parallel::sampling\_tag, 1238
  - \_\_gnu\_parallel::unbalanced\_tag, 1240
- \_\_glibcxx\_check\_erase
  - macros.h, 3387
- \_\_glibcxx\_check\_erase\_range
  - macros.h, 3387
- \_\_glibcxx\_check\_heap\_pred
  - macros.h, 3387



- 
- `__glibcxx_check_insert`  
  macros.h, 3387
  - `__glibcxx_check_insert_range`  
  macros.h, 3387
  - `__glibcxx_check_partitioned_lower`  
  macros.h, 3388
  - `__glibcxx_check_partitioned_lower_pred`  
  macros.h, 3388
  - `__glibcxx_check_partitioned_upper_pred`  
  macros.h, 3388
  - `__glibcxx_check_sorted_pred`  
  macros.h, 3388
  - `__gnu_cxx`, 339
    - `_Bit_scan_forward`, 362
    - `__static_pointer_cast`, 361
    - `operator<`, 365, 366
    - `operator<=`, 367, 368
    - `operator>`, 370
    - `operator>=`, 371, 372
    - `operator+`, 363–365
    - `operator==`, 368, 369
    - `swap`, 372
  - `__gnu_cxx::_Caster`, 849
  - `__gnu_cxx::_Char_types`, 849
  - `__gnu_cxx::_ExtPtr_allocator`, 850
  - `__gnu_cxx::_Invalid_type`, 852
  - `__gnu_cxx::_Pointer_adapter`, 852
  - `__gnu_cxx::_Relative_pointer_impl`, 855
  - `__gnu_cxx::_Relative_pointer_impl<const_Tp>`, 856
  - `__gnu_cxx::_Std_pointer_impl`, 856
  - `__gnu_cxx::_Unqualified_type`, 857
  - `__gnu_cxx::__common_pool_policy`, 768
  - `__gnu_cxx::__detail`, 373
    - `__bit_allocate`, 374
    - `__bit_free`, 374
    - `__num_bitmaps`, 374
    - `__num_blocks`, 375
  - `__gnu_cxx::__detail::_Bitmap_counter`, 769
  - `__gnu_cxx::__detail::_Ffit_finder`, 771
    - `argument_type`, 772
    - `result_type`, 772
  - `__gnu_cxx::__detail::_mini_vector`, 768
  - `__gnu_cxx::_mt_alloc`, 772
  - `__gnu_cxx::_mt_alloc_base`, 774
  - `__gnu_cxx::_per_type_pool_policy`, 775
  - `__gnu_cxx::_pool<false>`, 776
  - `__gnu_cxx::_pool<true>`, 777
  - `__gnu_cxx::_pool_alloc`, 779
  - `__gnu_cxx::_pool_alloc_base`, 781
  - `__gnu_cxx::_pool_base`, 782
  - `__gnu_cxx::_rc_string_base`, 784
  - `__gnu_cxx::_scoped_lock`, 787
  - `__gnu_cxx::_versa_string`, 787
    - `~__versa_string`, 796
    - `__versa_string`, 792–795
  - `append`, 796–798
  - `assign`, 799–802
  - `at`, 803
  - `back`, 804
  - `begin`, 804
  - `c_str`, 804
  - `capacity`, 805
  - `cbegin`, 805
  - `cend`, 805
  - `clear`, 805
  - `compare`, 806–809
  - `copy`, 810
  - `crbegin`, 810
  - `crend`, 811
  - `data`, 811
  - `empty`, 811
  - `end`, 811, 812
  - `erase`, 812, 813
  - `find`, 813–815
  - `find_first_not_of`, 815–817
  - `find_first_of`, 818, 819
  - `find_last_not_of`, 820, 821
  - `find_last_of`, 822, 823
  - `front`, 824
  - `get_allocator`, 824
  - `insert`, 825–829
  - `length`, 830
  - `max_size`, 830
  - `npos`, 848
  - `operator+=`, 830–832
  - `operator=`, 832, 833
  - `push_back`, 834
  - `rbegin`, 835
  - `rend`, 835
  - `replace`, 836–842
-

- reserve, 843
- resize, 844
- rfind, 844–846
- shrink\_to\_fit, 847
- size, 847
- substr, 847
- swap, 848
- \_\_gnu\_cxx::annotate\_base, 858
- \_\_gnu\_cxx::array\_allocator, 859
- \_\_gnu\_cxx::array\_allocator\_base, 860
- \_\_gnu\_cxx::binary\_compose, 861
  - argument\_type, 863
  - result\_type, 863
- \_\_gnu\_cxx::bitmap\_allocator, 863
  - \_M\_allocate\_single\_object, 865
  - \_M\_deallocate\_single\_object, 865
- \_\_gnu\_cxx::char\_traits, 866
- \_\_gnu\_cxx::character, 867
- \_\_gnu\_cxx::condition\_base, 868
- \_\_gnu\_cxx::constant\_binary\_fun, 869
- \_\_gnu\_cxx::constant\_unary\_fun, 870
- \_\_gnu\_cxx::constant\_void\_fun, 871
- \_\_gnu\_cxx::debug\_allocator, 872
- \_\_gnu\_cxx::enc\_filebuf, 873
  - \_M\_buf, 895
  - \_M\_buf\_locale, 895
  - \_M\_buf\_size, 895
  - \_M\_create\_pback, 878
  - \_M\_destroy\_pback, 878
  - \_M\_ext\_buf, 895
  - \_M\_ext\_buf\_size, 895
  - \_M\_ext\_next, 896
  - \_M\_in\_beg, 896
  - \_M\_in\_cur, 896
  - \_M\_in\_end, 896
  - \_M\_mode, 897
  - \_M\_out\_beg, 897
  - \_M\_out\_cur, 897
  - \_M\_out\_end, 898
  - \_M\_pback, 898
  - \_M\_pback\_cur\_save, 898
  - \_M\_pback\_end\_save, 898
  - \_M\_pback\_init, 899
  - \_M\_reading, 899
  - \_M\_set\_buffer, 878
  - \_\_streambuf\_type, 876
- char\_type, 877
- close, 879
- eback, 879
- egptr, 879
- epptr, 880
- gbump, 880
- getloc, 880
- gptr, 881
- imbue, 881
- in\_avail, 881
- int\_type, 877
- is\_open, 882
- off\_type, 877
- open, 882
- overflow, 883
- pbackfail, 883
- pbase, 884
- pbump, 884
- pos\_type, 877
- pptr, 885
- pubimbue, 885
- pubseekoff, 885
- pubseekpos, 886
- pubsetbuf, 886
- pubsync, 886
- sbumpc, 887
- seekoff, 887
- seekpos, 887
- setbuf, 888
- setg, 888
- setp, 889
- sgetc, 889
- sgetn, 889
- showmanyc, 890
- snextc, 890
- sputbackc, 891
- sputc, 891
- sputn, 891
- stossc, 892
- sungetc, 892
- sync, 892
- traits\_type, 878
- uflow, 893
- underflow, 893
- xsgetn, 894
- xsputn, 894

- 
- \_\_gnu\_cxx::encoding\_char\_traits, 899
  - \_\_gnu\_cxx::encoding\_state, 901
  - \_\_gnu\_cxx::forced\_error, 902
    - what, 903
  - \_\_gnu\_cxx::free\_list, 904
    - \_M\_clear, 905
    - \_M\_get, 905
    - \_M\_insert, 905
  - \_\_gnu\_cxx::hash\_map, 905
  - \_\_gnu\_cxx::hash\_multimap, 908
  - \_\_gnu\_cxx::hash\_multiset, 910
  - \_\_gnu\_cxx::hash\_set, 912
  - \_\_gnu\_cxx::limit\_condition, 914
  - \_\_gnu\_cxx::limit\_condition::always\_-
    - adjustor, 915
  - \_\_gnu\_cxx::limit\_condition::limit\_-
    - adjustor, 915
  - \_\_gnu\_cxx::limit\_condition::never\_-
    - adjustor, 916
  - \_\_gnu\_cxx::malloc\_allocator, 916
  - \_\_gnu\_cxx::new\_allocator, 917
  - \_\_gnu\_cxx::project1st, 919
    - first\_argument\_type, 920
    - result\_type, 920
    - second\_argument\_type, 920
  - \_\_gnu\_cxx::project2nd, 920
    - first\_argument\_type, 921
    - result\_type, 921
    - second\_argument\_type, 921
  - \_\_gnu\_cxx::random\_condition, 922
  - \_\_gnu\_cxx::random\_condition::always\_-
    - adjustor, 923
  - \_\_gnu\_cxx::random\_condition::group\_-
    - adjustor, 923
  - \_\_gnu\_cxx::random\_condition::never\_-
    - adjustor, 924
  - \_\_gnu\_cxx::rb\_tree, 924
  - \_\_gnu\_cxx::recursive\_init\_error, 927
    - what, 928
  - \_\_gnu\_cxx::rope, 929
  - \_\_gnu\_cxx::select1st, 935
    - argument\_type, 936
    - result\_type, 936
  - \_\_gnu\_cxx::select2nd, 936
    - argument\_type, 937
    - result\_type, 937
  - \_\_gnu\_cxx::slist, 937
  - \_\_gnu\_cxx::stdio\_filebuf, 940
    - ~stdio\_filebuf, 947
    - \_M\_buf, 968
    - \_M\_buf\_locale, 968
    - \_M\_buf\_size, 968
    - \_M\_create\_pback, 948
    - \_M\_destroy\_pback, 948
    - \_M\_ext\_buf, 969
    - \_M\_ext\_buf\_size, 969
    - \_M\_ext\_next, 969
    - \_M\_in\_beg, 969
    - \_M\_in\_cur, 970
    - \_M\_in\_end, 970
    - \_M\_mode, 970
    - \_M\_out\_beg, 971
    - \_M\_out\_cur, 971
    - \_M\_out\_end, 971
    - \_M\_pback, 972
    - \_M\_pback\_cur\_save, 972
    - \_M\_pback\_end\_save, 972
    - \_M\_pback\_init, 972
    - \_M\_reading, 973
    - \_M\_set\_buffer, 948
    - \_\_streambuf\_type, 945
  - char\_type, 945
  - close, 948
  - eback, 949
  - egptr, 949
  - epptr, 950
  - fd, 950
  - file, 950
  - gbump, 951
  - getloc, 951
  - gptr, 951
  - imbue, 952
  - in\_avail, 952
  - int\_type, 945
  - is\_open, 953
  - off\_type, 946
  - open, 953, 954
  - overflow, 954
  - pbackfail, 955
  - pbase, 955
  - pbump, 956
  - pos\_type, 946
-

- pptr, 956
- pubimbue, 957
- pubseekoff, 957
- pubseekpos, 957
- pubsetbuf, 958
- pubsync, 958
- sbumpc, 958
- seekoff, 959
- seekpos, 959
- setbuf, 960
- setg, 960
- setp, 961
- sgetc, 961
- sgetn, 962
- showmanyc, 962
- snextc, 963
- sputbackc, 963
- sputc, 963
- sputn, 964
- stdio\_filebuf, 946, 947
- stossc, 964
- sungetc, 965
- sync, 965
- traits\_type, 946
- uflow, 965
- underflow, 966
- xsgetn, 967
- xspn, 967
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 973
- \_\_M\_buf\_locale, 995
- \_\_M\_in\_beg, 995
- \_\_M\_in\_cur, 995
- \_\_M\_in\_end, 996
- \_\_M\_out\_beg, 996
- \_\_M\_out\_cur, 997
- \_\_M\_out\_end, 997
- \_\_streambuf\_type, 977
- char\_type, 977
- eback, 979
- egptr, 979
- epptr, 980
- file, 980
- gbump, 980
- getloc, 981
- gptr, 981
- imbue, 981
- in\_avail, 982
- int\_type, 978
- off\_type, 978
- overflow, 982
- pbackfail, 983
- pbase, 983
- pbump, 984
- pos\_type, 978
- pptr, 984
- pubimbue, 985
- pubseekoff, 985
- pubseekpos, 985
- pubsetbuf, 986
- pubsync, 986
- sbumpc, 986
- seekoff, 987
- seekpos, 987
- setbuf, 988
- setg, 988
- setp, 988
- sgetc, 989
- sgetn, 989
- showmanyc, 989
- snextc, 990
- sputbackc, 990
- sputc, 991
- sputn, 991
- stossc, 992
- sungetc, 992
- sync, 992
- traits\_type, 978
- uflow, 993
- underflow, 993
- xsgetn, 994
- xspn, 994
- \_\_gnu\_cxx::subtractive\_rng, 998
- argument\_type, 999
- operator(), 999
- result\_type, 999
- subtractive\_rng, 999
- \_\_gnu\_cxx::temporary\_buffer, 1000
- ~temporary\_buffer, 1001
- begin, 1002
- end, 1002
- requested\_size, 1002
- size, 1002

- 
- temporary\_buffer, 1001
  - \_\_gnu\_cxx::throw\_allocator\_base, 1003
  - \_\_gnu\_cxx::throw\_allocator\_limit, 1004
  - \_\_gnu\_cxx::throw\_allocator\_random, 1006
  - \_\_gnu\_cxx::throw\_value\_base, 1007
  - \_\_gnu\_cxx::throw\_value\_limit, 1008
  - \_\_gnu\_cxx::throw\_value\_random, 1010
  - \_\_gnu\_cxx::typelist, 375
  - apply\_generator, 375
  - \_\_gnu\_cxx::unary\_compose, 1011
  - argument\_type, 1013
  - result\_type, 1013
  - \_\_gnu\_debug, 376
    - \_check\_dereferenceable, 382
    - \_check\_singular, 383
    - \_check\_singular\_aux, 383
    - \_check\_string, 383
    - \_valid\_range, 384
    - \_valid\_range\_aux, 384
    - \_valid\_range\_aux2, 385
  - \_\_gnu\_debug::\_After\_nth\_from, 1014
  - \_\_gnu\_debug::\_Not\_equal\_to, 1014
  - \_\_gnu\_debug::\_Safe\_iterator, 1015
    - \_M\_attach, 1019
    - \_M\_attach\_single, 1019
    - \_M\_attached\_to, 1019
    - \_M\_can\_compare, 1019
    - \_M\_dereferenceable, 1020
    - \_M\_detach, 1020
    - \_M\_detach\_single, 1020
    - \_M\_get\_distance, 1020
    - \_M\_get\_mutex, 1020
    - \_M\_incrementable, 1021
    - \_M\_invalidate, 1021
    - \_M\_invalidate\_single, 1021
    - \_M\_is\_begin, 1022
    - \_M\_is\_end, 1022
    - \_M\_next, 1025
    - \_M\_prior, 1025
    - \_M\_sequence, 1025
    - \_M\_singular, 1022
    - \_M\_version, 1026
    - \_Safe\_iterator, 1017, 1018
  - base, 1022
  - operator\_iterator, 1023
  - operator\*, 1023
  - operator++, 1023
  - operator->, 1024
  - operator--, 1024
  - operator=, 1025
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 1026
    - \_M\_attach, 1029
    - \_M\_attach\_single, 1029
    - \_M\_attached\_to, 1029
    - \_M\_can\_compare, 1029
    - \_M\_detach, 1029
    - \_M\_detach\_single, 1029
    - \_M\_get\_mutex, 1029
    - \_M\_next, 1030
    - \_M\_prior, 1030
    - \_M\_sequence, 1030
    - \_M\_singular, 1030
    - \_M\_version, 1031
    - \_Safe\_iterator\_base, 1028
  - \_\_gnu\_debug::\_Safe\_sequence, 1031
    - \_M\_const\_iterators, 1034
    - \_M\_detach\_all, 1033
    - \_M\_detach\_singular, 1033
    - \_M\_get\_mutex, 1033
    - \_M\_invalidate\_all, 1033
    - \_M\_invalidate\_if, 1033
    - \_M\_iterators, 1034
    - \_M\_revalidate\_singular, 1033
    - \_M\_swap, 1034
    - \_M\_transfer\_iter, 1034
    - \_M\_version, 1035
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 1035
    - ~\_Safe\_sequence\_base, 1036
    - \_M\_const\_iterators, 1038
    - \_M\_detach\_all, 1037
    - \_M\_detach\_singular, 1037
    - \_M\_get\_mutex, 1037
    - \_M\_invalidate\_all, 1037
    - \_M\_iterators, 1038
    - \_M\_revalidate\_singular, 1037
    - \_M\_swap, 1037
    - \_M\_version, 1038
  - \_\_gnu\_debug::\_is\_same, 1013
  - \_\_gnu\_debug::basic\_string, 1038
    - \_M\_const\_iterators, 1090
-

- [\\_M\\_detach\\_all](#), 1047
- [\\_M\\_detach\\_singular](#), 1047
- [\\_M\\_get\\_mutex](#), 1047
- [\\_M\\_invalidate\\_all](#), 1047
- [\\_M\\_invalidate\\_if](#), 1047
- [\\_M\\_iterators](#), 1090
- [\\_M\\_revalidate\\_singular](#), 1048
- [\\_M\\_swap](#), 1048
- [\\_M\\_transfer\\_iter](#), 1048
- [\\_M\\_version](#), 1090
- [append](#), 1048–1050
- [assign](#), 1051–1053
- [at](#), 1054
- [back](#), 1055
- [begin](#), 1055
- [c\\_str](#), 1055
- [capacity](#), 1056
- [cbegin](#), 1056
- [cend](#), 1056
- [clear](#), 1056
- [compare](#), 1056–1059
- [copy](#), 1060
- [crbegin](#), 1060
- [crend](#), 1060
- [data](#), 1060
- [empty](#), 1061
- [end](#), 1061
- [erase](#), 1061, 1062
- [find](#), 1062–1064
- [find\\_first\\_not\\_of](#), 1064, 1065
- [find\\_first\\_of](#), 1066, 1067
- [find\\_last\\_not\\_of](#), 1067–1069
- [find\\_last\\_of](#), 1069, 1070
- [front](#), 1071
- [get\\_allocator](#), 1071
- [insert](#), 1071–1076
- [length](#), 1076
- [max\\_size](#), 1076
- [npos](#), 1091
- [operator+=](#), 1076, 1077
- [push\\_back](#), 1079
- [rbegin](#), 1079
- [rend](#), 1079
- [replace](#), 1080–1085
- [reserve](#), 1086
- [resize](#), 1086, 1087
- [rfind](#), 1087, 1088
- [shrink\\_to\\_fit](#), 1089
- [size](#), 1089
- [substr](#), 1089
- [swap](#), 1090
- [\\_\\_gnu\\_internal](#), 385
- [\\_\\_gnu\\_parallel](#), 385
- [\\_AlgorithmStrategy](#), 403
- [\\_BinIndex](#), 402
- [\\_CASable](#), 402
- [\\_CASable\\_bits](#), 450
- [\\_CASable\\_mask](#), 450
- [\\_FindAlgorithm](#), 403
- [\\_MultiwayMergeAlgorithm](#), 403
- [\\_Parallelism](#), 403
- [\\_PartialSumAlgorithm](#), 403
- [\\_SequenceIndex](#), 402
- [\\_SortAlgorithm](#), 403
- [\\_SplittingAlgorithm](#), 404
- [\\_ThreadIndex](#), 402
- [\\_\\_calc\\_borders](#), 404
- [\\_\\_compare\\_and\\_swap](#), 404
- [\\_\\_compare\\_and\\_swap\\_32](#), 405
- [\\_\\_compare\\_and\\_swap\\_64](#), 405
- [\\_\\_decode2](#), 405
- [\\_\\_determine\\_samples](#), 406
- [\\_\\_encode2](#), 406
- [\\_\\_fetch\\_and\\_add](#), 407
- [\\_\\_fetch\\_and\\_add\\_32](#), 407
- [\\_\\_fetch\\_and\\_add\\_64](#), 408
- [\\_\\_find\\_template](#), 408–410
- [\\_\\_for\\_each\\_template\\_random\\_access](#), 411
- [\\_\\_for\\_each\\_template\\_random\\_access\\_ed](#), 411
- [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop](#), 412
- [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\\_static](#), 413
- [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing](#), 413
- [\\_\\_is\\_sorted](#), 414
- [\\_\\_median\\_of\\_three\\_iterators](#), 415
- [\\_\\_merge\\_advance](#), 415
- [\\_\\_merge\\_advance\\_movc](#), 416
- [\\_\\_merge\\_advance\\_usual](#), 416

- 
- `__parallel_merge_advance`, 417, 418
  - `__parallel_nth_element`, 418
  - `__parallel_partial_sort`, 419
  - `__parallel_partial_sum`, 419
  - `__parallel_partial_sum_basecase`, 420
  - `__parallel_partial_sum_linear`, 420
  - `__parallel_partition`, 421
  - `__parallel_random_shuffle`, 422
  - `__parallel_random_shuffle_drs`, 422
  - `__parallel_random_shuffle_drs_pu`, 423
  - `__parallel_sort`, 423–427
  - `__parallel_sort_qs`, 428
  - `__parallel_sort_qs_conquer`, 428
  - `__parallel_sort_qs_divide`, 429
  - `__parallel_sort_qsb`, 429
  - `__parallel_unique_copy`, 430
  - `__qsb_conquer`, 431
  - `__qsb_divide`, 431
  - `__qsb_local_sort_with_helping`, 432
  - `__random_number_pow2`, 432
  - `__rd_log2`, 433
  - `__round_up_to_pow2`, 433
  - `__search_template`, 434
  - `__sequential_multiway_merge`, 434
  - `__sequential_random_shuffle`, 435
  - `__shrink`, 435
  - `__shrink_and_double`, 436
  - `__yield`, 436
  - `equally_split`, 436
  - `equally_split_point`, 437
  - `list_partition`, 437
  - `max`, 438
  - `min`, 438
  - `multiseq_partition`, 438
  - `multiseq_selection`, 439
  - `multiway_merge`, 440
  - `multiway_merge_3_variant`, 442
  - `multiway_merge_4_variant`, 442
  - `multiway_merge_exact_splitting`, 443
  - `multiway_merge_loser_tree`, 444
  - `multiway_merge_loser_tree_-sentinel`, 444
  - `multiway_merge_loser_tree_-unguarded`, 445
  - `multiway_merge_sampling_-splitting`, 446
  - `multiway_merge_sentinels`, 446
  - `parallel_multiway_merge`, 448
  - `parallel_sort_mwms`, 449
  - `parallel_sort_mwms_pu`, 449
  - `__gnu_parallel::_DRSSorterPU`, 1139
  - `_M_bins_begin`, 1140
  - `_M_num_threads`, 1140
  - `_M_sd`, 1140
  - `_M_seed`, 1141
  - `__bins_end`, 1140
  - `__gnu_parallel::_-DRandomShufflingGlobalData`, 1136
  - `_DRandomShufflingGlobalData`, 1137
  - `_M_bin_proc`, 1137
  - `_M_dist`, 1137
  - `_M_num_bins`, 1138
  - `_M_num_bits`, 1138
  - `_M_source`, 1138
  - `_M_starts`, 1138
  - `_M_temporaries`, 1139
  - `__gnu_parallel::_DummyReduct`, 1141
  - `__gnu_parallel::_EqualFromLess`, 1142
  - `first_argument_type`, 1143
  - `result_type`, 1143
  - `second_argument_type`, 1143
  - `__gnu_parallel::_EqualTo`, 1143
  - `first_argument_type`, 1145
  - `result_type`, 1145
  - `second_argument_type`, 1145
  - `__gnu_parallel::_GuardedIterator`, 1145
  - `_GuardedIterator`, 1146
  - `operator_RAIter`, 1146
  - `operator<`, 1147
  - `operator<=`, 1148
  - `operator*`, 1147
  - `operator++`, 1147
  - `__gnu_parallel::_IteratorPair`, 1148
  - `first`, 1150
  - `first_type`, 1150
  - `second`, 1150
-

- second\_type, 1150
- \_\_gnu\_parallel::\_IteratorTriple, 1151
- \_\_gnu\_parallel::\_Job, 1152
  - \_M\_first, 1153
  - \_M\_last, 1153
  - \_M\_load, 1153
- \_\_gnu\_parallel::\_Less, 1153
  - first\_argument\_type, 1155
  - result\_type, 1155
  - second\_argument\_type, 1155
- \_\_gnu\_parallel::\_Lexicographic, 1155
  - first\_argument\_type, 1157
  - result\_type, 1157
  - second\_argument\_type, 1157
- \_\_gnu\_parallel::\_LexicographicReverse, 1157
  - first\_argument\_type, 1159
  - result\_type, 1159
  - second\_argument\_type, 1159
- \_\_gnu\_parallel::\_LoserTree, 1159
  - \_M\_comp, 1162
  - \_M\_first\_insert, 1162
  - \_M\_log\_k, 1162
  - \_M\_losers, 1162
  - \_\_delete\_min\_insert, 1161
  - \_\_get\_min\_source, 1161
  - \_\_insert\_start, 1161
- \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1163
  - \_M\_comp, 1166
  - \_M\_first\_insert, 1166
  - \_M\_log\_k, 1166
  - \_M\_losers, 1166
  - \_\_delete\_min\_insert, 1164
  - \_\_get\_min\_source, 1164
  - \_\_init\_winner, 1165
  - \_\_insert\_start, 1165
- \_\_gnu\_parallel::\_LoserTreeBase, 1167
  - ~\_LoserTreeBase, 1168
  - \_LoserTreeBase, 1168
  - \_M\_comp, 1169
  - \_M\_first\_insert, 1169
  - \_M\_log\_k, 1170
  - \_M\_losers, 1170
  - \_\_get\_min\_source, 1169
  - \_\_insert\_start, 1169
- \_\_gnu\_parallel::\_LoserTreeBase::\_Loser, 1170
  - \_M\_key, 1171
  - \_M\_source, 1171
  - \_M\_sup, 1171
- \_\_gnu\_parallel::\_LoserTreePointer, 1172
- \_\_gnu\_parallel::\_LoserTreePointer< false, \_Tp, \_Compare >, 1173
- \_\_gnu\_parallel::\_LoserTreePointerBase, 1174
- \_\_gnu\_parallel::\_-LoserTreePointerBase::\_Loser, 1175
- \_\_gnu\_parallel::\_-LoserTreePointerUnguarded, 1176
- \_\_gnu\_parallel::\_-LoserTreePointerUnguarded< false, \_Tp, \_Compare >, 1177
- \_\_gnu\_parallel::\_-LoserTreePointerUnguardedBase, 1179
- \_\_gnu\_parallel::\_LoserTreeTraits, 1180
  - \_M\_use\_pointer, 1181
- \_\_gnu\_parallel::\_LoserTreeUnguarded, 1181
- \_\_gnu\_parallel::\_LoserTreeUnguarded< false, \_Tp, \_Compare >, 1183
- \_\_gnu\_parallel::\_-LoserTreeUnguardedBase, 1184
- \_\_gnu\_parallel::\_Multiplies, 1185
  - first\_argument\_type, 1187
  - result\_type, 1187
  - second\_argument\_type, 1187
- \_\_gnu\_parallel::\_Nothing, 1187
  - operator(), 1188
- \_\_gnu\_parallel::\_PMWMSortingData, 1191
  - \_M\_num\_threads, 1192
  - \_M\_offsets, 1192
  - \_M\_pieces, 1192
  - \_M\_samples, 1193
  - \_M\_source, 1193
  - \_M\_starts, 1193
  - \_M\_temporary, 1193



- 
- \_\_gnu\_parallel::\_Piece, 1188
  - \_M\_begin, 1189
  - \_M\_end, 1189
  - \_\_gnu\_parallel::\_Plus, 1189
  - first\_argument\_type, 1191
  - result\_type, 1191
  - second\_argument\_type, 1191
  - \_\_gnu\_parallel::\_PseudoSequence, 1194
  - PseudoSequence, 1194
  - begin, 1195
  - end, 1195
  - \_\_gnu\_parallel::\_-
    - PseudoSequenceIterator, 1195
  - \_\_gnu\_parallel::\_QSBThreadLocal, 1196
  - \_M\_elements\_leftover, 1198
  - \_M\_global, 1198
  - \_M\_initial, 1198
  - \_M\_leftover\_parts, 1198
  - \_M\_num\_threads, 1198
  - \_Piece, 1197
  - \_QSBThreadLocal, 1197
  - \_\_gnu\_parallel::\_RandomNumber, 1199
  - \_RandomNumber, 1199
  - \_genrand\_bits, 1200
  - operator(), 1200
  - \_\_gnu\_parallel::\_-
    - RestrictedBoundedConcurrentQueue, 1200
    - ~\_RestrictedBoundedConcurrentQueue, 1202
    - \_RestrictedBoundedConcurrentQueue, 1202
    - pop\_back, 1202
    - pop\_front, 1202
    - push\_front, 1202
  - \_\_gnu\_parallel::\_SamplingSorter, 1203
  - \_\_gnu\_parallel::\_SamplingSorter< false,
    - \_RAIter, \_StrictWeakOrdering >, 1204
  - \_\_gnu\_parallel::\_Settings, 1204
  - accumulate\_minimal\_n, 1206
  - adjacent\_difference\_minimal\_n, 1206
  - cache\_line\_size, 1207
  - count\_minimal\_n, 1207
  - fill\_minimal\_n, 1207
  - find\_increasing\_factor, 1207
  - find\_initial\_block\_size, 1207
  - find\_maximum\_block\_size, 1207
  - find\_scale\_factor, 1208
  - find\_sequential\_search\_size, 1208
  - for\_each\_minimal\_n, 1208
  - generate\_minimal\_n, 1208
  - get, 1206
  - L1\_cache\_size, 1208
  - L2\_cache\_size, 1208
  - max\_element\_minimal\_n, 1208
  - merge\_minimal\_n, 1209
  - merge\_oversampling, 1209
  - min\_element\_minimal\_n, 1209
  - multiway\_merge\_minimal\_k, 1209
  - multiway\_merge\_minimal\_n, 1209
  - multiway\_merge\_oversampling, 1209
  - nth\_element\_minimal\_n, 1210
  - partial\_sort\_minimal\_n, 1210
  - partial\_sum\_dilation, 1210
  - partial\_sum\_minimal\_n, 1210
  - partition\_chunk\_share, 1210
  - partition\_chunk\_size, 1210
  - partition\_minimal\_n, 1211
  - qsb\_steals, 1211
  - random\_shuffle\_minimal\_n, 1211
  - replace\_minimal\_n, 1211
  - search\_minimal\_n, 1211
  - set, 1206
  - set\_difference\_minimal\_n, 1211
  - set\_intersection\_minimal\_n, 1211
  - set\_symmetric\_difference\_minimal\_n, 1212
  - set\_union\_minimal\_n, 1212
  - sort\_minimal\_n, 1212
  - sort\_mwms\_oversampling, 1212
  - sort\_qs\_num\_samples\_preset, 1212
  - sort\_qsb\_base\_case\_maximal\_n, 1212
  - TLB\_size, 1213
  - transform\_minimal\_n, 1213
  - unique\_copy\_minimal\_n, 1213
  - \_\_gnu\_parallel::\_SplitConsistently, 1213
-

- 
- \_\_gnu\_parallel::\_\_SplitConsistently<  
false, \_RAIter, \_Compare,  
\_SortingPlacesIterator >, 1214
  - \_\_gnu\_parallel::\_\_SplitConsistently<  
true, \_RAIter, \_Compare,  
\_SortingPlacesIterator >, 1215
  - \_\_gnu\_parallel::\_\_accumulate\_binop\_  
reduct, 1091
  - \_\_gnu\_parallel::\_\_accumulate\_selector,  
1092
    - \_M\_finish\_iterator, 1093
    - operator(), 1093
  - \_\_gnu\_parallel::\_\_adjacent\_difference\_  
selector, 1093
    - \_M\_finish\_iterator, 1094
  - \_\_gnu\_parallel::\_\_adjacent\_find\_  
selector, 1095
    - \_M\_sequential\_algorithm, 1096
    - operator(), 1096
  - \_\_gnu\_parallel::\_\_binder1st, 1096
    - argument\_type, 1098
    - result\_type, 1098
  - \_\_gnu\_parallel::\_\_binder2nd, 1098
    - argument\_type, 1100
    - result\_type, 1100
  - \_\_gnu\_parallel::\_\_count\_if\_selector,  
1100
    - \_M\_finish\_iterator, 1101
    - operator(), 1101
  - \_\_gnu\_parallel::\_\_count\_selector, 1102
    - \_M\_finish\_iterator, 1103
    - operator(), 1103
  - \_\_gnu\_parallel::\_\_fill\_selector, 1104
    - \_M\_finish\_iterator, 1105
    - operator(), 1105
  - \_\_gnu\_parallel::\_\_find\_first\_of\_selector,  
1105
    - \_M\_sequential\_algorithm, 1107
    - operator(), 1107
  - \_\_gnu\_parallel::\_\_find\_if\_selector, 1107
    - \_M\_sequential\_algorithm, 1108
    - operator(), 1109
  - \_\_gnu\_parallel::\_\_for\_each\_selector,  
1109
    - \_M\_finish\_iterator, 1111
    - operator(), 1110
  - \_\_gnu\_parallel::\_\_generate\_selector,  
1111
    - \_M\_finish\_iterator, 1112
    - operator(), 1112
  - \_\_gnu\_parallel::\_\_generic\_find\_selector,  
1113
    - \_\_gnu\_parallel::\_\_generic\_for\_each\_  
selector, 1113
      - \_M\_finish\_iterator, 1115
    - \_\_gnu\_parallel::\_\_identity\_selector, 1115
      - \_M\_finish\_iterator, 1116
      - operator(), 1116
    - \_\_gnu\_parallel::\_\_inner\_product\_  
selector, 1117
      - \_M\_finish\_iterator, 1119
      - \_\_begin1\_iterator, 1118
      - \_\_begin2\_iterator, 1119
      - \_\_inner\_product\_selector, 1118
      - operator(), 1118
    - \_\_gnu\_parallel::\_\_max\_element\_reduct,  
1119
    - \_\_gnu\_parallel::\_\_min\_element\_reduct,  
1120
    - \_\_gnu\_parallel::\_\_mismatch\_selector,  
1121
      - \_M\_sequential\_algorithm, 1122
      - operator(), 1122
    - \_\_gnu\_parallel::\_\_multiway\_merge\_  
3\_variant\_sentinel\_switch,  
1122
    - \_\_gnu\_parallel::\_\_multiway\_merge\_  
3\_variant\_sentinel\_switch<  
true, \_RAIterIterator, \_RAIter3,  
\_DifferenceTp, \_Compare >,  
1123
    - \_\_gnu\_parallel::\_\_multiway\_merge\_  
4\_variant\_sentinel\_switch,  
1124
    - \_\_gnu\_parallel::\_\_multiway\_merge\_  
4\_variant\_sentinel\_switch<  
true, \_RAIterIterator, \_RAIter3,  
\_DifferenceTp, \_Compare >,  
1124
    - \_\_gnu\_parallel::\_\_multiway\_merge\_  
k\_variant\_sentinel\_switch,  
1125
-

- 
- \_\_gnu\_parallel::\_\_multiway\_merge\_-  
k\_variant\_sentinel\_switch<  
false, \_\_stable, \_RAIterIterator,  
\_RAIter3, \_DifferenceTp,  
\_Compare >, 1126
  - \_\_gnu\_parallel::\_\_replace\_if\_selector,  
1127
    - \_M\_finish\_iterator, 1128
    - \_new\_val, 1128
    - \_replace\_if\_selector, 1128
    - operator(), 1128
  - \_\_gnu\_parallel::\_\_replace\_selector, 1129
    - \_M\_finish\_iterator, 1131
    - \_new\_val, 1130
    - \_replace\_selector, 1130
    - operator(), 1130
  - \_\_gnu\_parallel::\_\_transform1\_selector,  
1131
    - \_M\_finish\_iterator, 1132
    - operator(), 1132
  - \_\_gnu\_parallel::\_\_transform2\_selector,  
1133
    - \_M\_finish\_iterator, 1134
    - operator(), 1134
  - \_\_gnu\_parallel::\_\_unary\_negate, 1134
    - argument\_type, 1136
    - result\_type, 1136
  - \_\_gnu\_parallel::balanced\_quicksort\_tag,  
1215
    - \_\_get\_num\_threads, 1216
    - set\_num\_threads, 1216
  - \_\_gnu\_parallel::balanced\_tag, 1217
    - \_\_get\_num\_threads, 1218
    - set\_num\_threads, 1218
  - \_\_gnu\_parallel::constant\_size\_blocks\_-  
tag, 1218
  - \_\_gnu\_parallel::default\_parallel\_tag,  
1219
    - \_\_get\_num\_threads, 1220
    - set\_num\_threads, 1220
  - \_\_gnu\_parallel::equal\_split\_tag, 1221
  - \_\_gnu\_parallel::exact\_tag, 1222
    - \_\_get\_num\_threads, 1223
    - set\_num\_threads, 1223
  - \_\_gnu\_parallel::find\_tag, 1223
  - \_\_gnu\_parallel::growing\_blocks\_tag,  
1224
  - \_\_gnu\_parallel::multiway\_mergesort\_-  
exact\_tag, 1225
    - \_\_get\_num\_threads, 1226
    - set\_num\_threads, 1226
  - \_\_gnu\_parallel::multiway\_mergesort\_-  
sampling\_tag, 1227
    - \_\_get\_num\_threads, 1228
    - set\_num\_threads, 1228
  - \_\_gnu\_parallel::multiway\_mergesort\_-  
tag, 1228
    - \_\_get\_num\_threads, 1229
    - set\_num\_threads, 1229
  - \_\_gnu\_parallel::omp\_loop\_static\_tag,  
1230
    - \_\_get\_num\_threads, 1231
    - set\_num\_threads, 1231
  - \_\_gnu\_parallel::omp\_loop\_tag, 1231
    - \_\_get\_num\_threads, 1232
    - set\_num\_threads, 1232
  - \_\_gnu\_parallel::parallel\_tag, 1233
    - \_\_get\_num\_threads, 1235
    - parallel\_tag, 1235
    - set\_num\_threads, 1235
  - \_\_gnu\_parallel::quicksort\_tag, 1236
    - \_\_get\_num\_threads, 1237
    - set\_num\_threads, 1237
  - \_\_gnu\_parallel::sampling\_tag, 1237
    - \_\_get\_num\_threads, 1238
    - set\_num\_threads, 1238
  - \_\_gnu\_parallel::sequential\_tag, 1239
  - \_\_gnu\_parallel::unbalanced\_tag, 1239
    - \_\_get\_num\_threads, 1240
    - set\_num\_threads, 1240
  - \_\_gnu\_pbds, 450
  - \_\_gnu\_pbds::associative\_container\_tag,  
1241
  - \_\_gnu\_pbds::basic\_hash\_table, 1242
  - \_\_gnu\_pbds::basic\_hash\_tag, 1243
  - \_\_gnu\_pbds::basic\_tree, 1244
  - \_\_gnu\_pbds::basic\_tree\_tag, 1245
  - \_\_gnu\_pbds::binary\_heap\_tag, 1246
  - \_\_gnu\_pbds::binomial\_heap\_tag, 1247
  - \_\_gnu\_pbds::cc\_hash\_table, 1248
  - \_\_gnu\_pbds::cc\_hash\_tag, 1250
-

- `__gnu_pbds::container_base`, 1251
- `__gnu_pbds::container_tag`, 1252
- `__gnu_pbds::container_traits`, 1253
- `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`, 1253
- `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >`, 1254
- `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >`, 1255
- `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >`, 1256
- `__gnu_pbds::gp_hash_table`, 1257
- `__gnu_pbds::gp_hash_tag`, 1259
- `__gnu_pbds::list_update`, 1260
- `__gnu_pbds::list_update_tag`, 1262
- `__gnu_pbds::null_mapped_type`, 1263
- `__gnu_pbds::ov_tree_tag`, 1264
- `__gnu_pbds::pairing_heap_tag`, 1265
- `__gnu_pbds::pat_trie_tag`, 1265
- `__gnu_pbds::priority_queue_tag`, 1267
- `__gnu_pbds::rb_tree_tag`, 1267
- `__gnu_pbds::rc_binomial_heap_tag`, 1269
- `__gnu_pbds::sequence_tag`, 1269
- `__gnu_pbds::splay_tree_tag`, 1270
- `__gnu_pbds::string_tag`, 1271
- `__gnu_pbds::thin_heap_tag`, 1272
- `__gnu_pbds::tree`, 1273
- `__gnu_pbds::tree_tag`, 1275
- `__gnu_pbds::trie`, 1276
- `__gnu_pbds::trie_tag`, 1278
- `__gnu_profile`, 453
  - `__env_t`, 459
  - `__get__global_lock`, 459
  - `__profcxx_init`, 459
  - `__report`, 459
- `__gnu_profile::__container_size_info`, 1279
- `__gnu_profile::__container_size_stack_info`, 1280
- `__gnu_profile::__hashfunc_info`, 1281
- `__gnu_profile::__hashfunc_stack_info`, 1283
- `__gnu_profile::__list2vector_info`, 1284
- `__gnu_profile::__map2umap_info`, 1285
- `__gnu_profile::__map2umap_stack_info`, 1287
- `__gnu_profile::__object_info_base`, 1288
- `__gnu_profile::__reentrance_guard`, 1289
- `__gnu_profile::__stack_hash`, 1290
- `__gnu_profile::__stack_info_base`, 1290
- `__gnu_profile::__trace_base`, 1291
- `__gnu_profile::__trace_container_size`, 1292
- `__gnu_profile::__trace_hash_func`, 1292
- `__gnu_profile::__trace_hashtable_size`, 1294
- `__gnu_profile::__trace_map2umap`, 1294
- `__gnu_profile::__trace_vector_size`, 1296
- `__gnu_profile::__trace_vector_to_list`, 1296
- `__gnu_profile::__vector2list_info`, 1298
- `__gnu_profile::__vector2list_stack_info`, 1299
- `__gnu_profile::__warning_data`, 1301
- `__gnu_sequential`, 460
- `__heap_select`
  - `std`, 602, 603
- `__init_winner`
  - `__gnu_parallel::__LoserTree< false, _Tp, _Compare >`, 1165
- `__inner_product_selector`
  - `__gnu_parallel::__inner_product_selector`, 1118
- `__inplace_stable_partition`
  - `std`, 603
- `__inplace_stable_sort`
  - `std`, 603
- `__insert_start`
  - `__gnu_parallel::__LoserTree`, 1161
  - `__gnu_parallel::__LoserTree< false, _Tp, _Compare >`, 1165
  - `__gnu_parallel::__LoserTreeBase`, 1169
- `__insertion_sort`
  - `std`, 604
- `__introsort_loop`

- 
- std, 604
  - \_\_invoke
    - std, 685
  - \_\_is\_sorted
    - \_\_gnu\_parallel, 414
  - \_\_iterator\_category
    - iterators, 271
  - \_\_lg
    - std, 605
  - \_\_match\_flag
    - std::regex\_constants, 744
  - \_\_median
    - SGIextensions, 16, 17
  - \_\_median\_of\_three\_iterators
    - \_\_gnu\_parallel, 415
  - \_\_merge\_adaptive
    - std, 605
  - \_\_merge\_advance
    - \_\_gnu\_parallel, 415
  - \_\_merge\_advance\_movc
    - \_\_gnu\_parallel, 416
  - \_\_merge\_advance\_usual
    - \_\_gnu\_parallel, 416
  - \_\_merge\_backward
    - std, 606
  - \_\_merge\_without\_buffer
    - std, 606
  - \_\_move\_median\_first
    - std, 607
  - \_\_new\_val
    - \_\_gnu\_parallel::\_\_replace\_if\_selector, 1128
    - \_\_gnu\_parallel::\_\_replace\_selector, 1130
  - \_\_num\_bitmaps
    - \_\_gnu\_cxx::\_\_detail, 374
  - \_\_num\_blocks
    - \_\_gnu\_cxx::\_\_detail, 375
  - \_\_num\_get\_type
    - std::basic\_fstream, 1547
    - std::basic\_ifstream, 1612
    - std::basic\_ios, 1663
    - std::basic\_iostream, 1697
    - std::basic\_istream, 1760
    - std::basic\_istreamstream, 1814
    - std::basic\_ofstream, 1867
    - std::basic\_ostream, 1910
    - std::basic\_ostringstream, 1955
    - std::basic\_stringstream, 2121
  - \_\_num\_put\_type
    - std::basic\_fstream, 1547, 1548
    - std::basic\_ifstream, 1612
    - std::basic\_ios, 1663
    - std::basic\_iostream, 1697, 1698
    - std::basic\_istream, 1761
    - std::basic\_istreamstream, 1814
    - std::basic\_ofstream, 1867
    - std::basic\_ostream, 1910
    - std::basic\_ostringstream, 1955
    - std::basic\_stringstream, 2121
  - \_\_parallel\_merge\_advance
    - \_\_gnu\_parallel, 417, 418
  - \_\_parallel\_nth\_element
    - \_\_gnu\_parallel, 418
  - \_\_parallel\_partial\_sort
    - \_\_gnu\_parallel, 419
  - \_\_parallel\_partial\_sum
    - \_\_gnu\_parallel, 419
  - \_\_parallel\_partial\_sum\_basecase
    - \_\_gnu\_parallel, 420
  - \_\_parallel\_partial\_sum\_linear
    - \_\_gnu\_parallel, 420
  - \_\_parallel\_partition
    - \_\_gnu\_parallel, 421
  - \_\_parallel\_random\_shuffle
    - \_\_gnu\_parallel, 422
  - \_\_parallel\_random\_shuffle\_drs
    - \_\_gnu\_parallel, 422
  - \_\_parallel\_random\_shuffle\_drs\_pu
    - \_\_gnu\_parallel, 423
  - \_\_parallel\_sort
    - \_\_gnu\_parallel, 423–427
  - \_\_parallel\_sort\_qs
    - \_\_gnu\_parallel, 428
  - \_\_parallel\_sort\_qs\_conquer
    - \_\_gnu\_parallel, 428
  - \_\_parallel\_sort\_qs\_divide
    - \_\_gnu\_parallel, 429
  - \_\_parallel\_sort\_qsb
    - \_\_gnu\_parallel, 429
  - \_\_parallel\_unique\_copy
    - \_\_gnu\_parallel, 430
-

- \_\_partition
  - std, 607
- \_\_profcxx\_init
  - \_\_gnu\_profile, 459
- \_\_qsb\_conquer
  - \_\_gnu\_parallel, 431
- \_\_qsb\_divide
  - \_\_gnu\_parallel, 431
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, 432
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, 432
- \_\_rd\_log2
  - \_\_gnu\_parallel, 433
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_\_replace\_if\_selector, 1128
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_\_replace\_selector, 1130
- \_\_report
  - \_\_gnu\_profile, 459
- \_\_reverse
  - std, 608
- \_\_rotate
  - std, 608, 609
- \_\_rotate\_adaptive
  - std, 609
- \_\_round\_up\_to\_pow2
  - \_\_gnu\_parallel, 433
- \_\_search\_n
  - std, 609, 610
- \_\_search\_template
  - \_\_gnu\_parallel, 434
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 434
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 435
- \_\_shrink
  - \_\_gnu\_parallel, 435
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 436
- \_\_stable\_partition\_adaptive
  - std, 610
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 361
- \_\_streambuf\_type
  - \_\_gnu\_cxx::enc\_filebuf, 876
  - \_\_gnu\_cxx::stdio\_filebuf, 945
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 977
  - std::basic\_filebuf, 1512
  - std::basic\_streambuf, 2007
  - std::basic\_stringbuf, 2090
- \_\_syntax\_option
  - std::regex\_constants, 744
- \_\_unguarded\_insertion\_sort
  - std, 610, 611
- \_\_unguarded\_linear\_insert
  - std, 611
- \_\_unguarded\_partition
  - std, 611, 612
- \_\_unguarded\_partition\_pivot
  - std, 612
- \_\_unique\_copy
  - std, 612–614
- \_\_valid\_range
  - \_\_gnu\_debug, 384
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 384
- \_\_valid\_range\_aux2
  - \_\_gnu\_debug, 385
- \_\_verbose\_terminate\_handler
  - exceptions, 36
- \_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string, 792–795
- \_\_yield
  - \_\_gnu\_parallel, 436
- a
  - std::extreme\_value\_distribution, 2426
  - std::weibull\_distribution, 3178
- abi, 460
- abs
  - complex\_numbers, 44
- accumulate
  - std, 615
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1206
- acos
  - complex\_numbers, 44

- 
- acosh
    - complex\_numbers, 44
  - Adaptors for pointers to functions, 257
  - Adaptors for pointers to members, 259
  - addressof
    - std, 615
  - adjacent\_difference
    - std, 616
  - adjacent\_difference\_minimal\_n
    - \_\_gnu\_parallel::\_Settings, 1206
  - adjacent\_find
    - non\_mutating\_algorithms, 151
  - adjustfield
    - std::basic\_fstream, 1599
    - std::basic\_ifstream, 1652
    - std::basic\_ios, 1683
    - std::basic\_iostream, 1748
    - std::basic\_istream, 1799
    - std::basic\_istreamstream, 1854
    - std::basic\_ofstream, 1898
    - std::basic\_ostream, 1940
    - std::basic\_ostreamstream, 1986
    - std::basic\_stringstream, 2172
    - std::ios\_base, 2563
  - advance
    - std, 617
  - algo.h, 3181
  - algbases.h, 3195
  - algorithm, 3197, 3199
  - algorithmfwd.h, 3199, 3206
  - Algorithms, 125
  - all
    - std::bitset, 2201
    - std::locale, 2638
  - all\_of
    - non\_mutating\_algorithms, 152
  - allocate\_shared
    - pointer\_abstractions, 68
    - std::shared\_ptr, 3012
  - allocator.h, 3219
  - allocator\_type
    - std::set, 2979
  - Allocators, 199
  - alpha
    - std::gamma\_distribution, 2481
  - any
    - std::bitset, 2201
  - any\_of
    - non\_mutating\_algorithms, 152
  - app
    - std::basic\_fstream, 1599
    - std::basic\_ifstream, 1653
    - std::basic\_ios, 1683
    - std::basic\_iostream, 1748
    - std::basic\_istream, 1799
    - std::basic\_istreamstream, 1854
    - std::basic\_ofstream, 1898
    - std::basic\_ostream, 1940
    - std::basic\_ostreamstream, 1986
    - std::basic\_stringstream, 2173
    - std::ios\_base, 2563
  - append
    - \_\_gnu\_cxx::\_\_versa\_string, 796–798
    - \_\_gnu\_debug::basic\_string, 1048–1050
    - std::basic\_string, 2037–2040
  - apply
    - numeric\_arrays, 90
  - apply\_generator
    - \_\_gnu\_cxx::typelist, 375
  - arg
    - complex\_numbers, 44
  - argument\_type
    - \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder, 772
    - \_\_gnu\_cxx::binary\_compose, 863
    - \_\_gnu\_cxx::select1st, 936
    - \_\_gnu\_cxx::select2nd, 937
    - \_\_gnu\_cxx::subtractive\_rng, 999
    - \_\_gnu\_cxx::unary\_compose, 1013
    - \_\_gnu\_parallel::\_\_binder1st, 1098
    - \_\_gnu\_parallel::\_\_binder2nd, 1100
    - \_\_gnu\_parallel::\_\_unary\_negate, 1136
    - std::\_Maybe\_unary\_or\_binary\_-function<\_Res, \_T1 >, 1449
    - std::binder1st, 2187
    - std::binder2nd, 2189
    - std::const\_mem\_fun\_ref\_t, 2279
    - std::const\_mem\_fun\_t, 2281
    - std::hash, 2500
-

- std::hash< \_\_debug::bitset< \_Nb >  
>, 2502
- std::hash< \_\_debug::vector< bool,  
\_Alloc > >, 2504
- std::hash< \_\_gnu\_cxx::throw\_-  
value\_limit >, 2506
- std::hash< \_\_gnu\_cxx::throw\_-  
value\_random >, 2508
- std::hash< \_\_profile::bitset< \_Nb >  
>, 2510
- std::hash< \_\_profile::vector< bool,  
\_Alloc > >, 2512
- std::hash< \_Tp \* >, 2514
- std::hash< error\_code >, 2516
- std::hash< shared\_ptr< \_Tp > >,  
2518
- std::hash< string >, 2520
- std::hash< thread::id >, 2522
- std::hash< u16string >, 2524
- std::hash< u32string >, 2526
- std::hash< unique\_ptr< \_Tp, \_Tp\_-  
Deleter > >, 2528
- std::hash< wstring >, 2530
- std::hash<::bitset< \_Nb > >, 2532
- std::hash<::vector< bool, \_Alloc >  
>, 2534
- std::logical\_not, 2649
- std::mem\_fun\_ref\_t, 2696
- std::mem\_fun\_t, 2698
- std::negate, 2789
- std::pointer\_to\_unary\_function,  
2917
- std::unary\_function, 3119
- std::unary\_negate, 3121
- Arithmetic Classes, 253
- array, 3219, 3220
- array\_allocator.h, 3221
- asin
  - complex\_numbers, 45
- asinh
  - complex\_numbers, 45
- assign
  - \_\_gnu\_cxx::\_\_versa\_string, 799–  
802
  - \_\_gnu\_debug::basic\_string, 1051–  
1053
- std::basic\_regex, 1997–2000
- std::basic\_string, 2040–2043
- std::deque, 2383, 2384
- std::forward\_list, 2441
- std::list, 2613, 2614
- std::vector, 3157
- assoc\_container.hpp, 3222
- assoc\_laguerre
  - tr1\_math\_spec\_func, 111
- assoc\_legendre
  - tr1\_math\_spec\_func, 111
- Associative, 29
- at
  - \_\_gnu\_cxx::\_\_versa\_string, 803
  - \_\_gnu\_debug::basic\_string, 1054
  - std::basic\_string, 2044
  - std::deque, 2384, 2385
  - std::map, 2662
  - std::vector, 3158
- atan
  - complex\_numbers, 45
- atanh
  - complex\_numbers, 45
- ate
  - std::basic\_fstream, 1599
  - std::basic\_ifstream, 1653
  - std::basic\_ios, 1683
  - std::basic\_iostream, 1748
  - std::basic\_istream, 1799
  - std::basic\_istreamstream, 1854
  - std::basic\_ofstream, 1898
  - std::basic\_ostream, 1941
  - std::basic\_ostreamstream, 1986
  - std::basic\_stringstream, 2173
  - std::ios\_base, 2563
- atomic, 3223
- atomic\_0.h, 3227
- atomic\_2.h, 3228
- atomic\_base.h, 3228
- atomic\_char
  - atomics, 207
- atomic\_char16\_t
  - atomics, 207
- atomic\_char32\_t
  - atomics, 207
- atomic\_int



- atomics, [207](#)
- atomic\_llong
  - atomics, [208](#)
- atomic\_long
  - atomics, [208](#)
- atomic\_schar
  - atomics, [208](#)
- atomic\_short
  - atomics, [208](#)
- atomic\_uchar
  - atomics, [208](#)
- atomic\_uint
  - atomics, [208](#)
- atomic\_ullong
  - atomics, [208](#)
- atomic\_ulong
  - atomics, [209](#)
- atomic\_ushort
  - atomics, [209](#)
- atomic\_wchar\_t
  - atomics, [209](#)
- atomic\_word.h, [3230](#)
- atomicfwd\_c.h, [3230](#)
- atomicfwd\_cxx.h, [3232](#)
- atomicity.h, [3232](#)
- Atoms, [201](#)
- atomics
  - \_GLIBCXX\_ATOMIC\_-  
PROPERTY, [207](#)
  - atomic\_char, [207](#)
  - atomic\_char16\_t, [207](#)
  - atomic\_char32\_t, [207](#)
  - atomic\_int, [207](#)
  - atomic\_llong, [208](#)
  - atomic\_long, [208](#)
  - atomic\_schar, [208](#)
  - atomic\_short, [208](#)
  - atomic\_uchar, [208](#)
  - atomic\_uint, [208](#)
  - atomic\_ullong, [208](#)
  - atomic\_ulong, [209](#)
  - atomic\_ushort, [209](#)
  - atomic\_wchar\_t, [209](#)
  - kill\_dependency, [209](#)
  - memory\_order, [209](#)
- auto\_ptr
  - std::auto\_ptr, [1494](#), [1495](#)
- auto\_ptr.h, [3233](#)
- awk
  - std::regex\_constants, [746](#)
- b
  - std::extreme\_value\_distribution,  
[2426](#)
  - std::weibull\_distribution, [3178](#)
- back
  - \_\_gnu\_cxx::\_\_versa\_string, [804](#)
  - \_\_gnu\_debug::basic\_string, [1055](#)
  - std::basic\_string, [2045](#)
  - std::deque, [2385](#), [2386](#)
  - std::list, [2614](#), [2615](#)
  - std::queue, [2931](#)
  - std::vector, [3159](#)
- back\_insert\_iterator
  - std::back\_insert\_iterator, [1501](#)
- back\_inserter
  - iterators, [271](#)
- bad
  - std::basic\_fstream, [1555](#)
  - std::basic\_ifstream, [1618](#)
  - std::basic\_ios, [1669](#)
  - std::basic\_iostream, [1705](#)
  - std::basic\_istream, [1766](#)
  - std::basic\_istreamstream, [1820](#)
  - std::basic\_ofstream, [1873](#)
  - std::basic\_ostream, [1916](#)
  - std::basic\_ostreamstream, [1961](#)
  - std::basic\_stringstream, [2129](#)
- badbit
  - std::basic\_fstream, [1600](#)
  - std::basic\_ifstream, [1653](#)
  - std::basic\_ios, [1684](#)
  - std::basic\_iostream, [1748](#)
  - std::basic\_istream, [1800](#)
  - std::basic\_istreamstream, [1855](#)
  - std::basic\_ofstream, [1899](#)
  - std::basic\_ostream, [1941](#)
  - std::basic\_ostreamstream, [1986](#)
  - std::basic\_stringstream, [2173](#)
  - std::ios\_base, [2564](#)
- balanced\_quicksort.h, [3234](#)
- base

- `__gnu_debug::_Safe_iterator`, 1022
  - `std::discard_block_engine`, 2402
  - `std::independent_bits_engine`, 2538
  - `std::reverse_iterator`, 2970
  - `std::shuffle_order_engine`, 3016
- `base.h`, 3235, 3237
- `basefield`
  - `std::basic_fstream`, 1600
  - `std::basic_ifstream`, 1653
  - `std::basic_ios`, 1684
  - `std::basic_iostream`, 1749
  - `std::basic_istream`, 1800
  - `std::basic_istreamstream`, 1855
  - `std::basic_ofstream`, 1899
  - `std::basic_ostream`, 1941
  - `std::basic_ostringstream`, 1986
  - `std::basic_stringstream`, 2173
  - `std::ios_base`, 2564
- `basic`
  - `std::regex_constants`, 746
- `basic_file.h`, 3237
- `basic_filebuf`
  - `std::basic_filebuf`, 1514
- `basic_fstream`
  - `std::basic_fstream`, 1553, 1554
- `basic_ifstream`
  - `std::basic_ifstream`, 1617
- `basic_ios`
  - `std::basic_ios`, 1668
- `basic_ios.h`, 3238
- `basic_ios.tcc`, 3238
- `basic_iostream`
  - `std::basic_iostream`, 1704
- `basic_istream`
  - `std::basic_istream`, 1765
- `basic_istreamstream`
  - `std::basic_istreamstream`, 1819
- `basic_iterator.h`, 3239
- `basic_ofstream`
  - `std::basic_ofstream`, 1871, 1872
- `basic_ostream`
  - `std::basic_ostream`, 1915
- `basic_ostringstream`
  - `std::basic_ostringstream`, 1959
- `basic_regex`
  - `std::basic_regex`, 1994–1996
- `basic_streambuf`
  - `std::basic_streambuf`, 2009
- `basic_string`
  - `std::basic_string`, 2034–2036
- `basic_string.h`, 3239
- `basic_string.tcc`, 3243
- `basic_stringbuf`
  - `std::basic_stringbuf`, 2092
- `basic_stringstream`
  - `std::basic_stringstream`, 2127, 2128
- `basic_types.hpp`, 3243
- `before_begin`
  - `std::forward_list`, 2442
- `beg`
  - `std::basic_fstream`, 1600
  - `std::basic_ifstream`, 1654
  - `std::basic_ios`, 1684
  - `std::basic_iostream`, 1749
  - `std::basic_istream`, 1800
  - `std::basic_istreamstream`, 1855
  - `std::basic_ofstream`, 1899
  - `std::basic_ostream`, 1941
  - `std::basic_ostringstream`, 1987
  - `std::basic_stringstream`, 2173
  - `std::ios_base`, 2564
- `begin`
  - `__gnu_cxx::__versa_string`, 804
  - `__gnu_cxx::temporary_buffer`, 1002
  - `__gnu_debug::basic_string`, 1055
  - `__gnu_parallel::_PseudoSequence`, 1195
  - `std::_Temporary_buffer`, 1464
  - `std::basic_string`, 2045
  - `std::deque`, 2386
  - `std::forward_list`, 2442
  - `std::list`, 2615
  - `std::map`, 2663
  - `std::match_results`, 2685
  - `std::multimap`, 2755
  - `std::multiset`, 2774
  - `std::set`, 2985
  - `std::vector`, 3159
- `Bernoulli`, 292
- `bernoulli_distribution`
  - `std::bernoulli_distribution`, 2180
- `beta`

- std::gamma\_distribution, 2481
- tr1\_math\_spec\_func, 111
- binary
  - std::basic\_fstream, 1600
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1684
  - std::basic\_istream, 1749
  - std::basic\_istream, 1800
  - std::basic\_istream, 1855
  - std::basic\_ofstream, 1899
  - std::basic\_ofstream, 1942
  - std::basic\_ostringstream, 1987
  - std::basic\_stringstream, 2174
  - std::ios\_base, 2564
- Binary Search, 193
- binary\_search
  - binary\_search\_algorithms, 195
- binary\_search\_algorithms
  - binary\_search, 195
  - equal\_range, 196
  - lower\_bound, 197
  - upper\_bound, 198, 199
- bind
  - binders, 124
  - std, 617
- bind1st
  - binders, 124
- bind2nd
  - binders, 124
- Binder Classes, 122
- binders
  - \_GLIBCXX\_DEPRECATED\_ATTR, 124
  - bind, 124
  - bind1st, 124
  - bind2nd, 124
- binders.h, 3244
- bitmap\_allocator.h, 3245
  - \_BALLOC\_ALIGN\_BYTES, 3246
- bitset, 3246, 3248, 3249
  - std::bitset, 2200, 2201
- boolalpha
  - std, 618
  - std::basic\_fstream, 1601
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1685
  - std::basic\_istream, 1749
  - std::basic\_istream, 1800
  - std::basic\_istream, 1856
  - std::basic\_ofstream, 1900
  - std::basic\_ofstream, 1942
  - std::basic\_ostringstream, 1987
  - std::basic\_stringstream, 2174
  - std::ios\_base, 2565
- Boolean Operations Classes, 255
- boost\_concept\_check.h, 3250
- boost\_sp\_counted\_base.h, 3251
- c
  - std::queue, 2933
- c++0x\_warning.h, 3251
- c++allocator.h, 3251
- c++config.h, 3252
- c++io.h, 3257
- c++locale.h, 3257
- c++locale\_internal.h, 3258
- c\_str
  - \_\_gnu\_cxx::\_\_versa\_string, 804
  - \_\_gnu\_debug::basic\_string, 1055
  - std::basic\_string, 2045
- cache\_line\_size
  - \_\_gnu\_parallel::\_Settings, 1207
- call\_once
  - mutexes, 71
  - std::once\_flag, 2885
- capacity
  - \_\_gnu\_cxx::\_\_versa\_string, 805
  - \_\_gnu\_debug::basic\_string, 1056
  - std::basic\_string, 2046
  - std::vector, 3160
- cassert, 3258
- category
  - std::locale, 2632
- cbefore\_begin
  - std::forward\_list, 2443
- cbegin
  - \_\_gnu\_cxx::\_\_versa\_string, 805
  - \_\_gnu\_debug::basic\_string, 1056
  - std::basic\_string, 2046
  - std::deque, 2386
  - std::forward\_list, 2443
  - std::list, 2615

- std::map, 2663
- std::match\_results, 2685
- std::multimap, 2755
- std::multiset, 2774
- std::set, 2985
- std::vector, 3160
- ccomplex, 3259
- cctype, 3259, 3260
- cend
  - \_\_gnu\_cxx::\_\_versa\_string, 805
  - \_\_gnu\_debug::basic\_string, 1056
  - std::basic\_string, 2046
  - std::deque, 2386
  - std::forward\_list, 2443
  - std::list, 2615
  - std::map, 2663
  - std::match\_results, 2685
  - std::multimap, 2755
  - std::multiset, 2775
  - std::set, 2985
  - std::vector, 3160
- cerr
  - std, 685
- cerno, 3260
- cfenv, 3260, 3261
- cfloat, 3261, 3262
- char\_traits.h, 3262
- char\_type
  - \_\_gnu\_cxx::enc\_filebuf, 877
  - \_\_gnu\_cxx::stdio\_filebuf, 945
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 977
  - std::\_\_ctype\_abstract\_base, 1311
  - std::basic\_filebuf, 1512
  - std::basic\_fstream, 1548
  - std::basic\_ifstream, 1613
  - std::basic\_ios, 1663
  - std::basic\_iostream, 1698
  - std::basic\_istream, 1761
  - std::basic\_istreamstream, 1815
  - std::basic\_ofstream, 1867
  - std::basic\_ostream, 1910
  - std::basic\_ostreamstream, 1955
  - std::basic\_streambuf, 2007
  - std::basic\_stringbuf, 2090
  - std::basic\_stringstream, 2122
  - std::collate, 2258
  - std::collate\_byname, 2265
  - std::ctype, 2284
  - std::ctype< char >, 2300
  - std::ctype< wchar\_t >, 2314
  - std::ctype\_byname, 2335
  - std::ctype\_byname< char >, 2350
  - std::istreambuf\_iterator, 2586
  - std::messages, 2701
  - std::messages\_byname, 2706
  - std::money\_get, 2714
  - std::money\_put, 2720
  - std::moneypunct, 2726
  - std::moneypunct\_byname, 2739
  - std::num\_get, 2806
  - std::num\_put, 2826
  - std::numput, 2872
  - std::numput\_byname, 2880
  - std::ostream\_iterator, 2886
  - std::ostreambuf\_iterator, 2891
  - std::time\_get, 3043
  - std::time\_get\_byname, 3055
  - std::time\_put, 3065
  - std::time\_put\_byname, 3070
- checkers.h, 3263
- chrono, 3263
- cin
  - std, 686
- cinttypes, 3267
- ciso646, 3267
- classic
  - std::locale, 2634
- classic\_table
  - std::ctype< char >, 2301
  - std::ctype\_byname< char >, 2351
- clear
  - \_\_gnu\_cxx::\_\_versa\_string, 805
  - \_\_gnu\_debug::basic\_string, 1056
  - std::basic\_fstream, 1556
  - std::basic\_ifstream, 1618
  - std::basic\_ios, 1669
  - std::basic\_iostream, 1705
  - std::basic\_istream, 1766
  - std::basic\_istreamstream, 1820
  - std::basic\_ofstream, 1873
  - std::basic\_ostream, 1916
  - std::basic\_ostreamstream, 1961

- std::basic\_string, 2046
- std::basic\_stringstream, 2129
- std::deque, 2387
- std::forward\_list, 2443
- std::list, 2615
- std::map, 2663
- std::multimap, 2755
- std::multiset, 2775
- std::set, 2985
- std::vector, 3160
- climits, 3268
- clocale, 3269
- clog
  - std, 686
- close
  - \_\_gnu\_cxx::enc\_filebuf, 879
  - \_\_gnu\_cxx::stdio\_filebuf, 948
  - std::basic\_filebuf, 1515
  - std::basic\_fstream, 1556
  - std::basic\_ifstream, 1619
  - std::basic\_ofstream, 1874
- cmath, 3269, 3272, 3276
- cmath.tcc, 3276
- code
  - std::regex\_error, 2949
- codecvt.h, 3276
- codecvt\_specializations.h, 3277
- collate
  - std::collate, 2258
  - std::locale, 2638
  - std::regex\_constants, 746
- combine
  - std::locale, 2634
- comp\_ellint\_1
  - tr1\_math\_spec\_func, 111
- comp\_ellint\_2
  - tr1\_math\_spec\_func, 112
- comp\_ellint\_3
  - tr1\_math\_spec\_func, 112
- compare
  - \_\_gnu\_cxx::\_\_versa\_string, 806–809
  - \_\_gnu\_debug::basic\_string, 1056–1059
  - std::basic\_string, 2047–2050
  - std::collate, 2259
  - std::collate\_byname, 2265
  - std::sub\_match, 3034
- Comparison Classes, 254
- compatibility.h, 3278
- compiletime\_settings.h, 3279
- \_GLIBCXX\_ASSERTIONS, 3280
- \_GLIBCXX\_CALL, 3280
- \_GLIBCXX\_RANDOM\_-SHUFFLE\_CONSIDER\_L1, 3280
- \_GLIBCXX\_RANDOM\_-SHUFFLE\_CONSIDER\_TLB, 3280
- \_GLIBCXX\_SCALE\_DOWN\_-FPU, 3281
- \_GLIBCXX\_VERBOSE\_LEVEL, 3281
- complex, 3281, 3285, 3286
  - std::complex, 2270
- Complex Numbers, 39
- complex.h, 3287
- complex\_numbers
  - abs, 44
  - acos, 44
  - acosh, 44
  - arg, 44
  - asin, 45
  - asinh, 45
  - atan, 45
  - atanh, 45
  - conj, 45
  - cos, 45
  - cosh, 45
  - exp, 46
  - fabs, 46
  - log, 46
  - log10, 46
  - norm, 46
  - operator<<, 50
  - operator>>, 52
  - operator\*, 47
  - operator\*=:, 48
  - operator+, 48
  - operator+=, 49
  - operator-, 49
  - operator-=, 49

- operator/, 50
- operator/=: 50
- operator=, 51
- operator==, 51
- polar, 52
- pow, 52
- sin, 53
- sinh, 53
- sqrt, 53
- tan, 53
- tanh, 53
- compose1
  - SGIextensions, 19
- compose2
  - SGIextensions, 19
- concept\_check.h, 3288
- concurrency.h, 3288
- Concurrency, 32
- cond\_dealtor.hpp, 3289
- Condition Variables, 54
- condition\_variable, 3289
- condition\_variables
  - cv\_status, 54
- conf\_hyperg
  - tr1\_math\_spec\_func, 112
- conj
  - complex\_numbers, 45
- const\_iterator
  - std::set, 2979
- const\_pointer
  - std::set, 2980
- const\_reference
  - std::set, 2980
- const\_reverse\_iterator
  - std::set, 2980
- constant0
  - SGIextensions, 19
- constant1
  - SGIextensions, 19
- constant2
  - SGIextensions, 19
- constructors\_destructor\_fn\_imps.hpp, 3290
- container\_base\_dispatch.hpp, 3291
- container\_type
  - std::back\_insert\_iterator, 1500
  - std::front\_insert\_iterator, 2460
  - std::insert\_iterator, 2547
- Containers, 26
- copy
  - \_\_gnu\_cxx::\_\_versa\_string, 810
  - \_\_gnu\_debug::basic\_string, 1060
  - mutating\_algorithms, 129
  - std::basic\_string, 2050
- copy\_backward
  - mutating\_algorithms, 129
- copy\_exception
  - exceptions, 36
- copy\_if
  - mutating\_algorithms, 130
- copy\_n
  - mutating\_algorithms, 130
  - SGIextensions, 20
- copyfmt
  - std::basic\_fstream, 1556
  - std::basic\_ifstream, 1619
  - std::basic\_ios, 1669
  - std::basic\_iostream, 1706
  - std::basic\_istream, 1767
  - std::basic\_istreamstream, 1821
  - std::basic\_ofstream, 1874
  - std::basic\_ostream, 1917
  - std::basic\_ostreamstream, 1962
  - std::basic\_stringstream, 2130
- cos
  - complex\_numbers, 45
- cosh
  - complex\_numbers, 45
- count
  - non\_mutating\_algorithms, 153
  - std::bitset, 2202
  - std::map, 2664
  - std::multimap, 2756
  - std::multiset, 2775
  - std::set, 2986
- count\_if
  - non\_mutating\_algorithms, 153
- count\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1207
- cout
  - std, 686
- cpp\_type\_traits.h, 3291

- cpu\_defines.h, 3292
- crbegin
  - \_\_gnu\_cxx::\_\_versa\_string, 810
  - \_\_gnu\_debug::basic\_string, 1060
  - std::basic\_string, 2051
  - std::deque, 2387
  - std::list, 2616
  - std::map, 2664
  - std::multimap, 2756
  - std::multiset, 2775
  - std::set, 2986
  - std::vector, 3161
- cref
  - std, 618
- cregex\_token\_iterator
  - regex, 220
- crend
  - \_\_gnu\_cxx::\_\_versa\_string, 811
  - \_\_gnu\_debug::basic\_string, 1060
  - std::basic\_string, 2051
  - std::deque, 2387
  - std::list, 2616
  - std::map, 2664
  - std::multimap, 2756
  - std::multiset, 2776
  - std::set, 2986
  - std::vector, 3161
- csetjmp, 3292
- cshift
  - numeric\_arrays, 91
- csignal, 3292
- cstdarg, 3293
- cstdbool, 3293, 3294
- cstddef, 3294
- cstdint, 3294, 3295
- cstdio, 3295, 3296
- cstdlib, 3296, 3297
- cstring, 3297
- csub\_match
  - regex, 220
- ctgmath, 3298
- ctime, 3298, 3299
- ctype
  - std::ctype< char >, 2300
  - std::ctype< wchar\_t >, 2315
  - std::locale, 2638
- ctype\_base.h, 3299
- ctype\_inline.h, 3300
- ctype\_noninline.h, 3300
- cur
  - std::basic\_fstream, 1601
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1685
  - std::basic\_iostream, 1750
  - std::basic\_istream, 1801
  - std::basic\_istreamstream, 1856
  - std::basic\_ofstream, 1900
  - std::basic\_ostream, 1942
  - std::basic\_ostreamstream, 1987
  - std::basic\_stringstream, 2174
  - std::ios\_base, 2565
- curr\_symbol
  - std::moneypunct, 2728
  - std::moneypunct\_byname, 2740
- current\_exception
  - exceptions, 36
- cv\_status
  - condition\_variables, 54
- cwchar, 3300, 3301
- cwctype, 3301, 3302
- cxxabi-forced.h, 3303
- cxxabi.h, 3303
- cxxabi\_tweaks.h, 3305
- cyl\_bessel\_i
  - tr1\_math\_spec\_func, 112
- cyl\_bessel\_j
  - tr1\_math\_spec\_func, 112
- cyl\_bessel\_k
  - tr1\_math\_spec\_func, 112
- cyl\_neumann
  - tr1\_math\_spec\_func, 113
- data
  - \_\_gnu\_cxx::\_\_versa\_string, 811
  - \_\_gnu\_debug::basic\_string, 1060
  - std::basic\_string, 2051
  - std::vector, 3161
- date\_order
  - std::time\_get, 3044
  - std::time\_get\_byname, 3055
- debug.h, 3305
- debug\_allocator.h, 3306

- debug\_map\_base.hpp, 3307
- dec
  - std, 618
  - std::basic\_fstream, 1601
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1685
  - std::basic\_istream, 1750
  - std::basic\_ostream, 1801
  - std::basic\_istream, 1856
  - std::basic\_ofstream, 1900
  - std::basic\_ostream, 1942
  - std::basic\_ostringstream, 1987
  - std::basic\_stringstream, 2174
  - std::ios\_base, 2565
- decimal, 3307
- Decimal Floating-Point Arithmetic, 121
- decimal128
  - std::decimal::decimal128, 2363
- decimal32
  - std::decimal::decimal32, 2365
- decimal32\_to\_long\_long
  - std::decimal, 741
- decimal64
  - std::decimal::decimal64, 2367
- decimal\_point
  - std::moneypunct, 2728
  - std::moneypunct\_byname, 2740
  - std::numpunct, 2873
  - std::numpunct\_byname, 2880
- denorm\_absent
  - std, 600
- denorm\_indeterminate
  - std, 600
- denorm\_present
  - std, 600
- denorm\_min
  - std::numeric\_limits, 2840
- densities
  - std::piecewise\_constant\_-distribution, 2903
  - std::piecewise\_linear\_distribution, 2908
- deque, 3319, 3320
  - std::deque, 2375–2377
- deque.tcc, 3321
- Diagnostics, 31
- difference\_type
  - std::back\_insert\_iterator, 1500
  - std::front\_insert\_iterator, 2460
  - std::insert\_iterator, 2547
  - std::istream\_iterator, 2583
  - std::istreambuf\_iterator, 2586
  - std::iterator, 2590
  - std::ostream\_iterator, 2886
  - std::ostreambuf\_iterator, 2891
  - std::raw\_storage\_iterator, 2944
  - std::reverse\_iterator, 2968
  - std::set, 2980
- digits
  - std::\_\_numeric\_limits\_base, 1395
  - std::numeric\_limits, 2841
- digits10
  - std::\_\_numeric\_limits\_base, 1395
  - std::numeric\_limits, 2842
- discard
  - std::discard\_block\_engine, 2402
  - std::independent\_bits\_engine, 2538
  - std::linear\_congruential\_engine, 2602
  - std::shuffle\_order\_engine, 3016
- discard\_block\_engine
  - std::discard\_block\_engine, 2400, 2401
- distance
  - SGIextensions, 20
  - std, 618
- do\_compare
  - std::collate, 2259
  - std::collate\_byname, 2265
- do\_curr\_symbol
  - std::moneypunct, 2728
  - std::moneypunct\_byname, 2740
- do\_date\_order
  - std::time\_get, 3045
  - std::time\_get\_byname, 3056
- do\_decimal\_point
  - std::moneypunct, 2729
  - std::moneypunct\_byname, 2741
  - std::numpunct, 2874
  - std::numpunct\_byname, 2880
- do\_falsename
  - std::numpunct, 2874



- std::numpunct\_byname, 2881
- do\_frac\_digits
  - std::moneypunct, 2729
  - std::moneypunct\_byname, 2741
- do\_get
  - std::money\_get, 2715
  - std::num\_get, 2807–2813
- do\_get\_date
  - std::time\_get, 3045
  - std::time\_get\_byname, 3056
- do\_get\_monthname
  - std::time\_get, 3046
  - std::time\_get\_byname, 3057
- do\_get\_time
  - std::time\_get, 3046
  - std::time\_get\_byname, 3057
- do\_get\_weekday
  - std::time\_get, 3047
  - std::time\_get\_byname, 3058
- do\_get\_year
  - std::time\_get, 3048
  - std::time\_get\_byname, 3059
- do\_grouping
  - std::moneypunct, 2730
  - std::moneypunct\_byname, 2741
  - std::numpunct, 2874
  - std::numpunct\_byname, 2881
- do\_hash
  - std::collate, 2260
  - std::collate\_byname, 2266
- do\_is
  - std::\_\_ctype\_abstract\_base, 1312
  - std::ctype, 2285
  - std::ctype< wchar\_t >, 2315, 2316
  - std::ctype\_byname, 2336
- do\_narrow
  - std::\_\_ctype\_abstract\_base, 1312, 1313
  - std::ctype, 2286
  - std::ctype< char >, 2301
  - std::ctype< wchar\_t >, 2317, 2318
  - std::ctype\_byname, 2336, 2337
  - std::ctype\_byname< char >, 2351
- do\_neg\_format
  - std::moneypunct, 2730
  - std::moneypunct\_byname, 2742
- do\_negative\_sign
  - std::moneypunct, 2730
  - std::moneypunct\_byname, 2742
- do\_out
  - std::\_\_codecvt\_abstract\_base, 1305
  - std::codecvt, 2229
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2235
  - std::codecvt< char, char, mbstate\_t >, 2240
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2245
  - std::codecvt\_byname, 2252
- do\_pos\_format
  - std::moneypunct, 2731
  - std::moneypunct\_byname, 2743
- do\_positive\_sign
  - std::moneypunct, 2731
  - std::moneypunct\_byname, 2743
- do\_put
  - std::money\_put, 2721
  - std::num\_put, 2827–2830
  - std::time\_put, 3066
  - std::time\_put\_byname, 3071
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base, 1314
  - std::ctype, 2287
  - std::ctype< wchar\_t >, 2319
  - std::ctype\_byname, 2338
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base, 1314
  - std::ctype, 2287
  - std::ctype< wchar\_t >, 2320
  - std::ctype\_byname, 2338
- do\_thousands\_sep
  - std::moneypunct, 2732
  - std::moneypunct\_byname, 2744
  - std::numpunct, 2875
  - std::numpunct\_byname, 2881
- do\_tolower
  - std::\_\_ctype\_abstract\_base, 1315
  - std::ctype, 2288
  - std::ctype< char >, 2302
  - std::ctype< wchar\_t >, 2321, 2322
  - std::ctype\_byname, 2339
  - std::ctype\_byname< char >, 2352

- do\_toupper
  - std::\_\_ctype\_abstract\_base, 1316
  - std::ctype, 2289
  - std::ctype< char >, 2303
  - std::ctype< wchar\_t >, 2323, 2324
  - std::ctype\_byname, 2340
  - std::ctype\_byname< char >, 2353
- do\_transform
  - std::collate, 2260
  - std::collate\_byname, 2266
- do\_truename
  - std::numpunct, 2875
  - std::numpunct\_byname, 2882
- do\_widen
  - std::\_\_ctype\_abstract\_base, 1317
  - std::ctype, 2289, 2290
  - std::ctype< char >, 2303, 2304
  - std::ctype< wchar\_t >, 2324, 2325
  - std::ctype\_byname, 2340, 2341
  - std::ctype\_byname< char >, 2353, 2354
- duration\_cast
  - std::chrono, 730
- eback
  - \_\_gnu\_cxx::enc\_filebuf, 879
  - \_\_gnu\_cxx::stdio\_filebuf, 949
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 979
  - std::basic\_filebuf, 1515
  - std::basic\_streambuf, 2010
  - std::basic\_stringbuf, 2092
- ECMAScript
  - std::regex\_constants, 747
- egptr
  - \_\_gnu\_cxx::enc\_filebuf, 879
  - \_\_gnu\_cxx::stdio\_filebuf, 949
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 979
  - std::basic\_filebuf, 1516
  - std::basic\_streambuf, 2010
  - std::basic\_stringbuf, 2093
- egrep
  - std::regex\_constants, 747
- element\_type
  - std::auto\_ptr, 1494
- ellint\_1
  - tr1\_math\_spec\_func, 113
- ellint\_2
  - tr1\_math\_spec\_func, 113
- ellint\_3
  - tr1\_math\_spec\_func, 113
- emplace
  - std::deque, 2387
  - std::list, 2616
  - std::vector, 3161
- emplace\_after
  - std::forward\_list, 2444
- emplace\_front
  - std::forward\_list, 2444
- empty
  - \_\_gnu\_cxx::\_\_versa\_string, 811
  - \_\_gnu\_debug::basic\_string, 1061
  - std::basic\_string, 2052
  - std::deque, 2388
  - std::forward\_list, 2444
  - std::list, 2617
  - std::map, 2665
  - std::match\_results, 2686
  - std::multimap, 2756
  - std::multiset, 2776
  - std::priority\_queue, 2925
  - std::queue, 2931
  - std::set, 2986
  - std::stack, 3025
  - std::vector, 3162
- enc\_filebuf.h, 3322
- end
  - \_\_gnu\_cxx::\_\_versa\_string, 811, 812
  - \_\_gnu\_cxx::temporary\_buffer, 1002
  - \_\_gnu\_debug::basic\_string, 1061
  - \_\_gnu\_parallel::\_PseudoSequence, 1195
  - std::\_Temporary\_buffer, 1464
  - std::basic\_fstream, 1601
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1685
  - std::basic\_iostream, 1750
  - std::basic\_istream, 1801
  - std::basic\_istream, 1856
  - std::basic\_ofstream, 1900
  - std::basic\_ostream, 1942
  - std::basic\_ostringstream, 1988

- std::basic\_string, 2052
- std::basic\_stringstream, 2174
- std::deque, 2388
- std::forward\_list, 2445
- std::ios\_base, 2565
- std::list, 2617
- std::map, 2665
- std::match\_results, 2686
- std::multimap, 2757
- std::multiset, 2776
- std::set, 2987
- std::vector, 3162
- endl
  - std, 619
- ends
  - std, 619
- eof
  - std::basic\_fstream, 1557
  - std::basic\_ifstream, 1620
  - std::basic\_ios, 1670
  - std::basic\_istream, 1706
  - std::basic\_istream, 1767
  - std::basic\_istream, 1821
  - std::basic\_ofstream, 1875
  - std::basic\_ostream, 1917
  - std::basic\_ostream, 1962
  - std::basic\_stringstream, 2130
- eofbit
  - std::basic\_fstream, 1601
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1685
  - std::basic\_istream, 1750
  - std::basic\_istream, 1801
  - std::basic\_istream, 1856
  - std::basic\_ofstream, 1900
  - std::basic\_ostream, 1942
  - std::basic\_ostream, 1988
  - std::basic\_stringstream, 2174
  - std::ios\_base, 2565
- epptr
  - \_\_gnu\_cxx::enc\_filebuf, 880
  - \_\_gnu\_cxx::stdio\_filebuf, 950
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 980
  - std::basic\_filebuf, 1516
  - std::basic\_streambuf, 2010
  - std::basic\_stringbuf, 2093
- epsilon
  - std::numeric\_limits, 2840
- equal
  - non\_mutating\_algorithms, 153, 154
  - std::istreambuf\_iterator, 2588
- equal\_range
  - binary\_search\_algorithms, 196
  - std::map, 2665, 2666
  - std::multimap, 2757, 2758
  - std::multiset, 2776, 2777
  - std::set, 2987
- equally\_split
  - \_\_gnu\_parallel, 436
- equally\_split.h, 3323
- equally\_split\_point
  - \_\_gnu\_parallel, 437
- erase
  - \_\_gnu\_cxx::\_\_versa\_string, 812, 813
  - \_\_gnu\_debug::basic\_string, 1061, 1062
  - std::basic\_string, 2052, 2053
  - std::deque, 2388, 2389
  - std::list, 2617, 2618
  - std::map, 2667
  - std::multimap, 2758, 2759
  - std::multiset, 2777, 2778
  - std::set, 2988, 2989
  - std::vector, 3162, 3163
- erase\_after
  - std::forward\_list, 2445, 2446
- error\_backref
  - std::regex\_constants, 744
- error\_badbrace
  - std::regex\_constants, 744
- error\_badrepeat
  - std::regex\_constants, 745
- error\_brace
  - std::regex\_constants, 745
- error\_brack
  - std::regex\_constants, 745
- error\_collate
  - std::regex\_constants, 745
- error\_complexity
  - std::regex\_constants, 745
- error\_constants.h, 3323

- error\_ctype
  - std::regex\_constants, 745
- error\_escape
  - std::regex\_constants, 745
- error\_paren
  - std::regex\_constants, 745
- error\_range
  - std::regex\_constants, 746
- error\_space
  - std::regex\_constants, 746
- error\_stack
  - std::regex\_constants, 746
- error\_type
  - std::regex\_constants, 744
- event
  - std::basic\_fstream, 1553
  - std::basic\_ifstream, 1616
  - std::basic\_ios, 1668
  - std::basic\_iostream, 1703
  - std::basic\_istream, 1765
  - std::basic\_istream, 1818
  - std::basic\_ofstream, 1871
  - std::basic\_ostream, 1915
  - std::basic\_ostream, 1959
  - std::basic\_stringstream, 2127
  - std::ios\_base, 2557
- event\_callback
  - std::basic\_fstream, 1548
  - std::basic\_ifstream, 1613
  - std::basic\_ios, 1664
  - std::basic\_iostream, 1699
  - std::basic\_istream, 1761
  - std::basic\_istream, 1815
  - std::basic\_ofstream, 1867
  - std::basic\_ostream, 1911
  - std::basic\_ostream, 1955
  - std::basic\_stringstream, 2122
  - std::ios\_base, 2554
- exception, 3324
- exception.hpp, 3325
- exception\_ptr.h, 3326
- Exceptions, 33
- exceptions
  - \_\_verbose\_terminate\_handler, 36
  - copy\_exception, 36
  - current\_exception, 36
  - make\_exception\_ptr, 36
  - rethrow\_exception, 36
  - rethrow\_if\_nested, 37
  - set\_terminate, 37
  - set\_unexpected, 37
  - std::basic\_fstream, 1557
  - std::basic\_ifstream, 1620
  - std::basic\_ios, 1670, 1671
  - std::basic\_iostream, 1706, 1707
  - std::basic\_istream, 1767, 1768
  - std::basic\_istream, 1822
  - std::basic\_ofstream, 1875
  - std::basic\_ostream, 1918
  - std::basic\_ostringstream, 1963
  - std::basic\_stringstream, 2131
  - terminate, 37
  - terminate\_handler, 35
  - throw\_with\_nested, 37
  - uncaught\_exception, 37
  - unexpected, 38
  - unexpected\_handler, 35
- exp
  - complex\_numbers, 46
- expint
  - tr1\_math\_spec\_func, 113
- exponential\_distribution
  - std::exponential\_distribution, 2422
- extended
  - std::regex\_constants, 747
- Extensions, 11
- extptr\_allocator.h, 3327
- fabs
  - complex\_numbers, 46
- facet
  - std::locale::facet, 2641
- fail
  - std::basic\_fstream, 1558
  - std::basic\_ifstream, 1621
  - std::basic\_ios, 1671
  - std::basic\_iostream, 1707
  - std::basic\_istream, 1768
  - std::basic\_istream, 1823
  - std::basic\_ofstream, 1876
  - std::basic\_ostream, 1919
  - std::basic\_ostringstream, 1964

- std::basic\_stringstream, 2132
- failbit
  - std::basic\_fstream, 1602
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1686
  - std::basic\_iostream, 1750
  - std::basic\_istream, 1801
  - std::basic\_istream, 1857
  - std::basic\_ofstream, 1901
  - std::basic\_ostream, 1943
  - std::basic\_ostringstream, 1988
  - std::basic\_stringstream, 2175
  - std::ios\_base, 2566
- failed
  - std::ostreambuf\_iterator, 2893
- false\_type
  - metaprogramming, 121
- falsename
  - std::numpunct, 2876
  - std::numpunct\_byname, 2882
- fd
  - \_\_gnu\_cxx::stdio\_filebuf, 950
- features.h, 3327
  - \_GLIBCXX\_BAL\_QUICKSORT, 3328
  - \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS, 3328
  - \_GLIBCXX\_FIND\_EQUAL\_SPLIT, 3328
  - \_GLIBCXX\_FIND\_GROWING\_BLOCKS, 3328
  - \_GLIBCXX\_MERGESORT, 3329
  - \_GLIBCXX\_QUICKSORT, 3329
  - \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING, 3329
  - \_GLIBCXX\_TREE\_FULL\_COPY, 3329
  - \_GLIBCXX\_TREE\_INITIAL\_SPLITTING, 3330
- fenv.h, 3330
- file
  - \_\_gnu\_cxx::stdio\_filebuf, 950
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 980
- filebuf
  - io, 61
- fill
  - mutating\_algorithms, 131
  - std::basic\_fstream, 1559
  - std::basic\_ifstream, 1621, 1622
  - std::basic\_ios, 1672
  - std::basic\_iostream, 1708
  - std::basic\_istream, 1769
  - std::basic\_istream, 1823, 1824
  - std::basic\_ofstream, 1876, 1877
  - std::basic\_ostream, 1919, 1920
  - std::basic\_ostringstream, 1964
  - std::basic\_stringstream, 2132, 2133
- fill\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1207
- fill\_n
  - mutating\_algorithms, 131
- find
  - \_\_gnu\_cxx::\_\_versa\_string, 813–815
  - \_\_gnu\_debug::basic\_string, 1062–1064
  - non\_mutating\_algorithms, 154
  - std::basic\_string, 2054, 2055
  - std::map, 2668
  - std::multimap, 2760
  - std::multiset, 2779
  - std::set, 2989, 2990
- find.h, 3330
- find\_end
  - non\_mutating\_algorithms, 155, 156
- find\_first\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 815–817
  - \_\_gnu\_debug::basic\_string, 1064, 1065
  - std::basic\_string, 2056, 2057
- find\_first\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 818, 819
  - \_\_gnu\_debug::basic\_string, 1066, 1067
  - non\_mutating\_algorithms, 156, 157
  - std::basic\_string, 2058, 2059
- find\_if
  - non\_mutating\_algorithms, 158
- find\_if\_not
  - non\_mutating\_algorithms, 158

- find\_increasing\_factor
  - \_\_gnu\_parallel::\_Settings, 1207
- find\_initial\_block\_size
  - \_\_gnu\_parallel::\_Settings, 1207
- find\_last\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 820, 821
  - \_\_gnu\_debug::basic\_string, 1067–1069
  - std::basic\_string, 2060, 2061
- find\_last\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 822, 823
  - \_\_gnu\_debug::basic\_string, 1069, 1070
  - std::basic\_string, 2062, 2063
- find\_maximum\_block\_size
  - \_\_gnu\_parallel::\_Settings, 1207
- find\_scale\_factor
  - \_\_gnu\_parallel::\_Settings, 1208
- find\_selectors.h, 3331
- find\_sequential\_search\_size
  - \_\_gnu\_parallel::\_Settings, 1208
- first
  - \_\_gnu\_parallel::\_IteratorPair, 1150
  - std::pair, 2901
  - std::sub\_match, 3036
- first\_argument\_type
  - \_\_gnu\_cxx::project1st, 920
  - \_\_gnu\_cxx::project2nd, 921
  - \_\_gnu\_parallel::\_EqualFromLess, 1143
  - \_\_gnu\_parallel::\_EqualTo, 1145
  - \_\_gnu\_parallel::\_Less, 1155
  - \_\_gnu\_parallel::\_Lexicographic, 1157
  - \_\_gnu\_parallel::\_-
    - LexicographicReverse, 1159
  - \_\_gnu\_parallel::\_Multiplies, 1187
  - \_\_gnu\_parallel::\_Plus, 1191
  - std::\_Maybe\_unary\_or\_binary\_-
    - function< \_Res, \_T1, \_T2 >, 1451
  - std::binary\_function, 2184
  - std::binary\_negate, 2185
  - std::const\_mem\_fun1\_ref\_t, 2275
  - std::const\_mem\_fun1\_t, 2277
  - std::divides, 2411
  - std::equal\_to, 2416
  - std::greater, 2490
  - std::greater\_equal, 2492
  - std::less, 2597
  - std::less\_equal, 2599
  - std::logical\_and, 2647
  - std::logical\_or, 2651
  - std::mem\_fun1\_ref\_t, 2692
  - std::mem\_fun1\_t, 2694
  - std::minus, 2708
  - std::modulus, 2710
  - std::multiplies, 2769
  - std::not\_equal\_to, 2803
  - std::plus, 2913
  - std::pointer\_to\_binary\_function, 2915
- first\_type
  - \_\_gnu\_parallel::\_IteratorPair, 1150
  - std::pair, 2900
  - std::sub\_match, 3033
- fixed
  - std, 620
  - std::basic\_fstream, 1602
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1686
  - std::basic\_iostream, 1751
  - std::basic\_istream, 1802
  - std::basic\_istream, 1857
  - std::basic\_ofstream, 1901
  - std::basic\_ostream, 1943
  - std::basic\_ostringstream, 1988
  - std::basic\_stringstream, 2175
  - std::ios\_base, 2566
- flags
  - std::basic\_fstream, 1559, 1560
  - std::basic\_ifstream, 1622
  - std::basic\_ios, 1672, 1673
  - std::basic\_iostream, 1709
  - std::basic\_istream, 1770
  - std::basic\_istream, 1824
  - std::basic\_ofstream, 1877
  - std::basic\_ostream, 1920
  - std::basic\_ostringstream, 1965
  - std::basic\_regex, 2000

- std::basic\_stringstream, 2133
- std::ios\_base, 2557, 2558
- flip
  - std::bitset, 2202
- float\_denorm\_style
  - std, 600
- float\_round\_style
  - std, 600
- floatfield
  - std::basic\_fstream, 1602
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1686
  - std::basic\_istream, 1751
  - std::basic\_istream, 1802
  - std::basic\_istream, 1857
  - std::basic\_ofstream, 1901
  - std::basic\_ostream, 1943
  - std::basic\_ostringstream, 1988
  - std::basic\_stringstream, 2175
  - std::ios\_base, 2566
- flush
  - std, 620
  - std::basic\_fstream, 1560
  - std::basic\_istream, 1709
  - std::basic\_ofstream, 1878
  - std::basic\_ostream, 1921
  - std::basic\_ostringstream, 1966
  - std::basic\_stringstream, 2134
- fmtflags
  - std::basic\_fstream, 1549
  - std::basic\_ifstream, 1613
  - std::basic\_ios, 1664
  - std::basic\_istream, 1699
  - std::basic\_istream, 1762
  - std::basic\_istream, 1815
  - std::basic\_ofstream, 1868
  - std::basic\_ostream, 1911
  - std::basic\_ostringstream, 1956
  - std::basic\_stringstream, 2123
  - std::ios\_base, 2554
- for\_each
  - non\_mutating\_algorithms, 158
- for\_each.h, 3332
- for\_each\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1208
- for\_each\_selectors.h, 3333
- format
  - std::match\_results, 2686, 2687
- format\_default
  - std::regex\_constants, 747
- format\_first\_only
  - std::regex\_constants, 748
- format\_no\_copy
  - std::regex\_constants, 748
- format\_sed
  - std::regex\_constants, 748
- formatter.h, 3335
- forward
  - std, 620
- forward\_list
  - std::forward\_list, 2438–2440
- forward\_list.h, 3336
- forward\_list.tcc, 3337
- fpos
  - std::fpos, 2457
- frac\_digits
  - std::moneypunct, 2732
  - std::moneypunct\_byname, 2744
- front
  - \_\_gnu\_cxx::\_\_versa\_string, 824
  - \_\_gnu\_debug::basic\_string, 1071
  - std::basic\_string, 2063, 2064
  - std::deque, 2389, 2390
  - std::forward\_list, 2446
  - std::list, 2618
  - std::queue, 2931, 2932
  - std::vector, 3163, 3164
- front\_insert\_iterator
  - std::front\_insert\_iterator, 2461
- front\_inserter
  - iterators, 271
- fstream, 3338
  - io, 61
- fstream.tcc, 3339
- functexcept.h, 3339
- function
  - std::function<\_Res(\_ArgTypes...)>, 2465, 2466
- Function Objects, 251
- functional, 3340, 3344
- functional\_hash.h, 3345
- functions.h, 3346

- functors
  - mem\_fn, 253
- future, 3348
  - std::future, 2472
  - std::future< \_Res & >, 2475
  - std::future< void >, 2477
- future\_category
  - futures, 57
- future\_errc
  - futures, 57
- Futures, 55
- futures
  - future\_category, 57
  - future\_errc, 57
- gamma\_distribution
  - std::gamma\_distribution, 2481
- gbump
  - \_\_gnu\_cxx::enc\_filebuf, 880
  - \_\_gnu\_cxx::stdio\_filebuf, 951
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 980
  - std::basic\_filebuf, 1516
  - std::basic\_streambuf, 2011
  - std::basic\_stringbuf, 2093
- gcount
  - std::basic\_fstream, 1560
  - std::basic\_ifstream, 1623
  - std::basic\_iostream, 1710
  - std::basic\_istream, 1770
  - std::basic\_istream, 1825
  - std::basic\_stringstream, 2134
- generate
  - mutating\_algorithms, 132
- generate\_canonical
  - random, 214
- generate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1208
- generate\_n
  - mutating\_algorithms, 132
- get
  - \_\_gnu\_parallel::Settings, 1206
  - std::auto\_ptr, 1496
  - std::basic\_fstream, 1561–1564
  - std::basic\_ifstream, 1623–1626
  - std::basic\_iostream, 1710–1713
  - std::basic\_istream, 1771–1773
  - std::basic\_istream, 1825–1828
  - std::basic\_stringstream, 2134–2137
  - std::future, 2472
  - std::future< \_Res & >, 2475
  - std::future< void >, 2477
  - std::money\_get, 2716, 2717
  - std::num\_get, 2814–2822
  - std::shared\_future, 2999
  - std::shared\_future< \_Res & >, 3002
- get\_allocator
  - \_\_gnu\_cxx::\_\_versa\_string, 824
  - \_\_gnu\_debug::basic\_string, 1071
  - std::basic\_string, 2064
  - std::deque, 2390
  - std::forward\_list, 2446
  - std::list, 2619
  - std::map, 2669
  - std::match\_results, 2687
  - std::multimap, 2760
  - std::multiset, 2780
  - std::set, 2990
- get\_date
  - std::time\_get, 3048
  - std::time\_get\_byname, 3059
- get\_deleter
  - pointer\_abstractions, 68
- get\_id
  - std::this\_thread, 753
- get\_money
  - std, 621
- get\_monthname
  - std::time\_get, 3049
  - std::time\_get\_byname, 3060
- get\_temporary\_buffer
  - std, 621
- get\_time
  - std::time\_get, 3050
  - std::time\_get\_byname, 3061
- get\_weekday
  - std::time\_get, 3050
  - std::time\_get\_byname, 3061
- get\_year
  - std::time\_get, 3051
  - std::time\_get\_byname, 3062
- getline
  - std, 621–623



- std::basic\_fstream, 1564
- std::basic\_ifstream, 1626, 1627
- std::basic\_iostream, 1713, 1714
- std::basic\_istream, 1774, 1775
- std::basic\_istreamstream, 1828, 1829
- std::basic\_stringstream, 2138, 2139
- getloc
  - \_\_gnu\_cxx::enc\_filebuf, 880
  - \_\_gnu\_cxx::stdio\_filebuf, 951
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 981
  - std::basic\_filebuf, 1517
  - std::basic\_fstream, 1565
  - std::basic\_ifstream, 1628
  - std::basic\_ios, 1673
  - std::basic\_iostream, 1715
  - std::basic\_istream, 1775
  - std::basic\_istreamstream, 1830
  - std::basic\_ofstream, 1878
  - std::basic\_ostream, 1921
  - std::basic\_ostringstream, 1966
  - std::basic\_regex, 2000
  - std::basic\_streambuf, 2011
  - std::basic\_stringbuf, 2094
  - std::basic\_stringstream, 2139
  - std::ios\_base, 2558
  - std::regex\_traits, 2961
- global
  - std::locale, 2635
- good
  - std::basic\_fstream, 1566
  - std::basic\_ifstream, 1628
  - std::basic\_ios, 1673
  - std::basic\_iostream, 1715
  - std::basic\_istream, 1775
  - std::basic\_istreamstream, 1830
  - std::basic\_ofstream, 1878
  - std::basic\_ostream, 1921
  - std::basic\_ostringstream, 1966
  - std::basic\_stringstream, 2139
- goodbit
  - std::basic\_fstream, 1602
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1686
  - std::basic\_iostream, 1751
  - std::basic\_istream, 1802
  - std::basic\_istreamstream, 1857
- std::basic\_ofstream, 1901
- std::basic\_ostream, 1943
- std::basic\_ostringstream, 1989
- std::basic\_stringstream, 2175
- std::ios\_base, 2566
- gptr
  - \_\_gnu\_cxx::enc\_filebuf, 881
  - \_\_gnu\_cxx::stdio\_filebuf, 951
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 981
  - std::basic\_filebuf, 1517
  - std::basic\_streambuf, 2011
  - std::basic\_stringbuf, 2094
- grep
  - std::regex\_constants, 748
- grouping
  - std::moneypunct, 2732
  - std::moneypunct\_byname, 2744
  - std::numpunct, 2876
  - std::numpunct\_byname, 2882
- gslice
  - numeric\_arrays, 87
- gslice.h, 3351
- gslice\_array
  - numeric\_arrays, 87
- gslice\_array.h, 3351
- has\_denorm
  - std::\_\_numeric\_limits\_base, 1395
  - std::numeric\_limits, 2842
- has\_denorm\_loss
  - std::\_\_numeric\_limits\_base, 1396
  - std::numeric\_limits, 2842
- has\_facet
  - std, 624
  - std::locale, 2637
  - std::locale::id, 2643
- has\_infinity
  - std::\_\_numeric\_limits\_base, 1396
  - std::numeric\_limits, 2842
- has\_quiet\_NaN
  - std::\_\_numeric\_limits\_base, 1396
  - std::numeric\_limits, 2842
- has\_signaling\_NaN
  - std::\_\_numeric\_limits\_base, 1396
  - std::numeric\_limits, 2842
- hash

- std::collate, 2261
- std::collate\_byname, 2267
- hash\_fun.h, 3352
- hash\_map, 3352
- hash\_policy.hpp, 3353
- hash\_set, 3354
- Hashes, 210
- hashtable.h, 3355, 3356
- hashtable\_policy.h, 3357
- Heap, 260
- heap\_algorithms
  - is\_heap, 262
  - is\_heap\_until, 262, 263
  - make\_heap, 263, 264
  - pop\_heap, 264
  - push\_heap, 265
  - sort\_heap, 266
- hermite
  - tr1\_math\_spec\_func, 114
- hex
  - std, 624
  - std::basic\_fstream, 1603
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1687
  - std::basic\_iostream, 1751
  - std::basic\_istream, 1802
  - std::basic\_istream, 1858
  - std::basic\_ofstream, 1902
  - std::basic\_ostream, 1944
  - std::basic\_ostringstream, 1989
  - std::basic\_stringstream, 2176
  - std::ios\_base, 2567
- hours
  - std::chrono, 729
- hyperg
  - tr1\_math\_spec\_func, 114
- I/O, 57
- icase
  - std::regex\_constants, 749
- id
  - std::collate, 2262
  - std::collate\_byname, 2268
  - std::ctype, 2296
  - std::ctype< char >, 2310
  - std::ctype< wchar\_t >, 2331
  - std::ctype\_byname, 2347
  - std::ctype\_byname< char >, 2360
  - std::locale::id, 2642
  - std::messages, 2702
  - std::messages\_byname, 2706
  - std::money\_get, 2717
  - std::money\_put, 2723
  - std::moneypunct, 2736
  - std::moneypunct\_byname, 2748
  - std::num\_get, 2823
  - std::num\_put, 2838
  - std::numput, 2877
  - std::numput\_byname, 2884
  - std::time\_get, 3052
  - std::time\_get\_byname, 3063
  - std::time\_put, 3068
  - std::time\_put\_byname, 3073
- identity\_element
  - SGIextensions, 20, 21
- ifstream
  - io, 61
- ignore
  - std::basic\_fstream, 1566, 1567
  - std::basic\_ifstream, 1628–1630
  - std::basic\_iostream, 1715–1717
  - std::basic\_istream, 1776, 1777
  - std::basic\_istream, 1830, 1831
  - std::basic\_stringstream, 2140, 2141
- imbue
  - \_\_gnu\_cxx::enc\_filebuf, 881
  - \_\_gnu\_cxx::stdio\_filebuf, 952
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 981
  - std::basic\_filebuf, 1517
  - std::basic\_fstream, 1568
  - std::basic\_ifstream, 1630
  - std::basic\_ios, 1674
  - std::basic\_iostream, 1717
  - std::basic\_istream, 1777
  - std::basic\_istream, 1832
  - std::basic\_ofstream, 1879
  - std::basic\_ostream, 1922
  - std::basic\_ostringstream, 1967
  - std::basic\_regex, 2001
  - std::basic\_streambuf, 2012
  - std::basic\_stringbuf, 2094
  - std::basic\_stringstream, 2141

- std::ios\_base, 2558
- std::regex\_traits, 2961
- in
  - std::\_\_codecvt\_abstract\_base, 1305
  - std::basic\_fstream, 1603
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1687
  - std::basic\_iostream, 1752
  - std::basic\_istream, 1803
  - std::basic\_istreamstream, 1858
  - std::basic\_ofstream, 1902
  - std::basic\_ostream, 1944
  - std::basic\_ostringstream, 1989
  - std::basic\_stringstream, 2176
  - std::codecvt, 2229
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2235
  - std::codecvt< char, char, mbstate\_t >, 2240
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2245
  - std::codecvt\_byname, 2252
  - std::ios\_base, 2567
- in\_avail
  - \_\_gnu\_cxx::enc\_filebuf, 881
  - \_\_gnu\_cxx::stdio\_filebuf, 952
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 982
  - std::basic\_filebuf, 1518
  - std::basic\_streambuf, 2012
  - std::basic\_stringbuf, 2095
- include/ Directory Reference, 324
- include/backward/ Directory Reference, 308
- include/bits/ Directory Reference, 311
- include/debug/ Directory Reference, 314
- include/decimal/ Directory Reference, 315
- include/ext/ Directory Reference, 319
- include/ext/pb\_ds/ Directory Reference, 332
- include/ext/pb\_ds/detail/ Directory Reference, 316
- include/parallel/ Directory Reference, 329
- include/profile/ Directory Reference, 334
- include/profile/impl/ Directory Reference, 322
- include/tr1/ Directory Reference, 336
- include/tr1\_impl/ Directory Reference, 337
- include/x86\_64-unknown-linux-gnu/ Directory Reference, 338
- include/x86\_64-unknown-linux-gnu/bits/ Directory Reference, 309
- includes
  - set\_algorithms, 187
- increment
  - std::linear\_congruential\_engine, 2605
- independent\_bits\_engine
  - std::independent\_bits\_engine, 2536, 2537
- indirect\_array
  - numeric\_arrays, 87
- indirect\_array.h, 3358
- infinity
  - std::numeric\_limits, 2840
- init
  - std::basic\_fstream, 1568
  - std::basic\_ifstream, 1631
  - std::basic\_ios, 1674
  - std::basic\_iostream, 1718
  - std::basic\_istream, 1778
  - std::basic\_istreamstream, 1832
  - std::basic\_ofstream, 1879
  - std::basic\_ostream, 1922
  - std::basic\_ostringstream, 1967
  - std::basic\_stringstream, 2142
- initializer\_list, 3358
- inner\_product
  - std, 624, 625
- inplace\_merge
  - sorting\_algorithms, 167
- insert
  - \_\_gnu\_cxx::\_\_versa\_string, 825–829
  - \_\_gnu\_debug::basic\_string, 1071–1076
  - std::basic\_string, 2064–2068
  - std::deque, 2390–2392
  - std::list, 2619–2621
  - std::map, 2669–2671
  - std::multimap, 2761, 2762

- std::multiset, 2780, 2781
- std::set, 2990–2992
- std::vector, 3164–3166
- insert\_after
  - std::forward\_list, 2447, 2448
- insert\_iterator
  - std::insert\_iterator, 2548
- inserter
  - iterators, 272
- int\_type
  - \_\_gnu\_cxx::enc\_filebuf, 877
  - \_\_gnu\_cxx::stdio\_filebuf, 945
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 978
  - std::basic\_filebuf, 1512
  - std::basic\_fstream, 1550
  - std::basic\_ifstream, 1614
  - std::basic\_ios, 1665
  - std::basic\_iostream, 1700
  - std::basic\_istream, 1762
  - std::basic\_istreamstream, 1816
  - std::basic\_ofstream, 1869
  - std::basic\_ostream, 1912
  - std::basic\_ostringstream, 1957
  - std::basic\_streambuf, 2007
  - std::basic\_stringbuf, 2090
  - std::basic\_stringstream, 2123, 2124
  - std::istreambuf\_iterator, 2586
- internal
  - std, 626
  - std::basic\_fstream, 1603
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1687
  - std::basic\_iostream, 1752
  - std::basic\_istream, 1803
  - std::basic\_istreamstream, 1858
  - std::basic\_ofstream, 1902
  - std::basic\_ostream, 1944
  - std::basic\_ostringstream, 1989
  - std::basic\_stringstream, 2176
  - std::ios\_base, 2567
- intervals
  - std::piecewise\_constant\_-distribution, 2903
  - std::piecewise\_linear\_distribution, 2908
- intl
  - std::moneypunct, 2736
  - std::moneypunct\_byname, 2748
- io
  - filebuf, 61
  - fstream, 61
  - ifstream, 61
  - ios, 61
  - iostream, 61
  - istream, 61
  - istreamstream, 61
  - ofstream, 61
  - ostream, 62
  - ostringstream, 62
  - streambuf, 62
  - stringbuf, 62
  - stringstream, 62
  - wfilebuf, 62
  - wfstream, 62
  - wifstream, 63
  - wios, 63
  - wiostream, 63
  - wistream, 63
  - wstringstream, 63
  - wofstream, 63
  - wostream, 63
  - wstringstream, 64
  - wstreambuf, 64
  - wstringbuf, 64
  - wstringstream, 64
- iomanip, 3359
- ios, 3360
  - io, 61
- ios\_base.h, 3360
- iosfwd, 3363
- iostate
  - std::basic\_fstream, 1550
  - std::basic\_ifstream, 1614
  - std::basic\_ios, 1665
  - std::basic\_iostream, 1700
  - std::basic\_istream, 1763
  - std::basic\_istreamstream, 1816
  - std::basic\_ofstream, 1869
  - std::basic\_ostream, 1912
  - std::basic\_ostringstream, 1957
  - std::basic\_stringstream, 2124
  - std::ios\_base, 2555

- 
- iostream, [3364](#)
    - io, [61](#)
  - iota
    - SGIextensions, [21](#)
    - std, [626](#)
  - is
    - std::\_\_ctype\_abstract\_base, [1318](#)
    - std::ctype, [2290](#), [2291](#)
    - std::ctype< char >, [2304](#), [2305](#)
    - std::ctype< wchar\_t >, [2326](#)
    - std::ctype\_byname, [2341](#), [2342](#)
    - std::ctype\_byname< char >, [2354](#), [2355](#)
  - is\_bounded
    - std::\_\_numeric\_limits\_base, [1396](#)
    - std::numeric\_limits, [2842](#)
  - is\_exact
    - std::\_\_numeric\_limits\_base, [1396](#)
    - std::numeric\_limits, [2843](#)
  - is\_heap
    - heap\_algorithms, [262](#)
    - SGIextensions, [21](#)
  - is\_heap\_until
    - heap\_algorithms, [262](#), [263](#)
  - is\_iec559
    - std::\_\_numeric\_limits\_base, [1397](#)
    - std::numeric\_limits, [2843](#)
  - is\_integer
    - std::\_\_numeric\_limits\_base, [1397](#)
    - std::numeric\_limits, [2843](#)
  - is\_modulo
    - std::\_\_numeric\_limits\_base, [1397](#)
    - std::numeric\_limits, [2843](#)
  - is\_open
    - \_\_gnu\_cxx::enc\_filebuf, [882](#)
    - \_\_gnu\_cxx::stdio\_filebuf, [953](#)
    - std::basic\_filebuf, [1518](#)
    - std::basic\_fstream, [1568](#)
    - std::basic\_ifstream, [1631](#)
    - std::basic\_ofstream, [1879](#)
  - is\_partitioned
    - mutating\_algorithms, [133](#)
  - is\_signed
    - std::\_\_numeric\_limits\_base, [1397](#)
    - std::numeric\_limits, [2843](#)
  - is\_sorted
    - SGIextensions, [22](#)
    - sorting\_algorithms, [168](#)
  - is\_sorted\_until
    - sorting\_algorithms, [169](#)
  - is\_specialized
    - std::\_\_numeric\_limits\_base, [1397](#)
    - std::numeric\_limits, [2844](#)
  - isalnum
    - std, [626](#)
  - isalpha
    - std, [626](#)
  - isctrl
    - std, [626](#)
  - isctype
    - regex, [222](#)
  - isdigit
    - std, [627](#)
  - isgraph
    - std, [627](#)
  - islower
    - std, [627](#)
  - isprint
    - std, [627](#)
  - ispunct
    - std, [627](#)
  - isspace
    - std, [627](#)
  - istream, [3365](#)
    - io, [61](#)
  - istream.tcc, [3366](#)
  - istream\_iterator
    - std::istream\_iterator, [2584](#)
  - istream\_type
    - std::istreambuf\_iterator, [2586](#)
  - istreambuf\_iterator
    - std::istreambuf\_iterator, [2587](#), [2588](#)
  - istreamstring
    - io, [61](#)
  - isupper
    - std, [627](#)
  - isxdigit
    - std, [627](#)
  - iter\_swap
    - mutating\_algorithms, [133](#)
  - iter\_type
    - std::money\_get, [2714](#)
-

- std::money\_put, 2720
- std::num\_get, 2806
- std::num\_put, 2826
- std::time\_get, 3044
- std::time\_get\_byname, 3055
- std::time\_put, 3065
- std::time\_put\_byname, 3070
- iterator, 3367
  - std::set, 2980
- iterator.h, 3367
- iterator\_category
  - std::back\_insert\_iterator, 1500
  - std::front\_insert\_iterator, 2460
  - std::insert\_iterator, 2547
  - std::istream\_iterator, 2583
  - std::istreambuf\_iterator, 2586
  - std::iterator, 2590
  - std::ostream\_iterator, 2887
  - std::ostreambuf\_iterator, 2891
  - std::raw\_storage\_iterator, 2944
  - std::reverse\_iterator, 2968
- iterator\_tracker.h, 3368
- Iterators, 267
- iterators
  - \_\_iterator\_category, 271
  - back\_inserter, 271
  - front\_inserter, 271
  - inserter, 272
  - operator==, 272, 273
- iword
  - std::basic\_fstream, 1569
  - std::basic\_ifstream, 1631
  - std::basic\_ios, 1674
  - std::basic\_iostream, 1718
  - std::basic\_istream, 1778
  - std::basic\_istreamstream, 1833
  - std::basic\_ofstream, 1880
  - std::basic\_ostream, 1922
  - std::basic\_ostreamstream, 1967
  - std::basic\_stringstream, 2142
  - std::ios\_base, 2559
- k
  - std::negative\_binomial\_distribution, 2791
- key\_comp
  - std::map, 2671
  - std::multimap, 2762
  - std::multiset, 2782
  - std::set, 2992
- key\_compare
  - std::set, 2981
- key\_type
  - std::set, 2981
- kill\_dependency
  - atomics, 209
- L1\_cache\_size
  - \_\_gnu\_parallel::\_Settings, 1208
- L2\_cache\_size
  - \_\_gnu\_parallel::\_Settings, 1208
- laguerre
  - tr1\_math\_spec\_func, 114
- lambda
  - std::exponential\_distribution, 2422
- left
  - std, 628
  - std::basic\_fstream, 1603
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1687
  - std::basic\_iostream, 1752
  - std::basic\_istream, 1803
  - std::basic\_istreamstream, 1858
  - std::basic\_ofstream, 1902
  - std::basic\_ostream, 1944
  - std::basic\_ostreamstream, 1990
  - std::basic\_stringstream, 2176
  - std::ios\_base, 2567
- legendre
  - tr1\_math\_spec\_func, 114
- length
  - \_\_gnu\_cxx::\_\_versa\_string, 830
  - \_\_gnu\_debug::basic\_string, 1076
  - std::basic\_string, 2069
  - std::match\_results, 2687
  - std::regex\_traits, 2962
  - std::sub\_match, 3035
- lexicographical\_compare
  - sorting\_algorithms, 170
- lexicographical\_compare\_3way
  - SGIextensions, 22
- limits, 3370

- 
- linear\_congruential\_engine
    - std::linear\_congruential\_engine, 2601
  - list, 3372–3374
    - std::list, 2610–2612
  - list.tcc, 3375
  - list\_partition
    - \_\_gnu\_parallel, 437
  - list\_partition.h, 3375
  - list\_update\_policy.hpp, 3376
  - locale, 3376
    - std::locale, 2632–2634
  - locale\_classes.h, 3376
  - locale\_classes.tcc, 3377
  - locale\_facets.h, 3378
  - locale\_facets.tcc, 3381
  - locale\_facets\_nonio.h, 3381
  - locale\_facets\_nonio.tcc, 3383
  - localefwd.h, 3383
  - Locales, 210
  - lock
    - mutexes, 71
  - log
    - complex\_numbers, 46
  - log10
    - complex\_numbers, 46
  - logic\_error
    - std::logic\_error, 2645
  - lookup\_classname
    - std::regex\_traits, 2962
  - lookup\_collatename
    - std::regex\_traits, 2963
  - losertree.h, 3384
  - lower\_bound
    - binary\_search\_algorithms, 197
    - std::map, 2671, 2672
    - std::multimap, 2763
    - std::multiset, 2782
    - std::set, 2992, 2993
  - lowest
    - std::numeric\_limits, 2840
  - macros.h, 3386
    - \_GLIBCXX\_DEBUG\_VERIFY, 3388
    - \_\_glibcxx\_check\_erase, 3387
    - \_\_glibcxx\_check\_erase\_range, 3387
    - \_\_glibcxx\_check\_heap\_pred, 3387
    - \_\_glibcxx\_check\_insert, 3387
    - \_\_glibcxx\_check\_insert\_range, 3387
    - \_\_glibcxx\_check\_partitioned\_lower, 3388
    - \_\_glibcxx\_check\_partitioned\_lower\_pred, 3388
    - \_\_glibcxx\_check\_partitioned\_upper\_pred, 3388
    - \_\_glibcxx\_check\_sorted\_pred, 3388
  - make\_exception\_ptr
    - exceptions, 36
  - make\_heap
    - heap\_algorithms, 263, 264
  - make\_pair
    - std, 628
  - make\_shared
    - pointer\_abstractions, 68
  - malloc\_allocator.h, 3389
  - map, 3389, 3390
    - std::map, 2660–2662
  - map.h, 3390, 3391
  - mark\_count
    - std::basic\_regex, 2001
  - mask\_array
    - numeric\_arrays, 88
  - mask\_array.h, 3392
  - match\_any
    - std::regex\_constants, 749
  - match\_continuous
    - std::regex\_constants, 749
  - match\_default
    - std::regex\_constants, 749
  - match\_flag\_type
    - std::regex\_constants, 743
  - match\_not\_bol
    - std::regex\_constants, 749
  - match\_not\_bow
    - std::regex\_constants, 749
  - match\_not\_eol
    - std::regex\_constants, 750
  - match\_not\_eow
    - std::regex\_constants, 750
  - match\_not\_null
    - std::regex\_constants, 750
-

- match\_prev\_avail
  - std::regex\_constants, 750
- match\_results
  - std::match\_results, 2684
- Mathematical Special Functions, 108
- max
  - \_\_gnu\_parallel, 438
  - numeric\_arrays, 91
  - sorting\_algorithms, 171
  - std::bernoulli\_distribution, 2180
  - std::binomial\_distribution, 2191
  - std::cauchy\_distribution, 2210
  - std::chi\_squared\_distribution, 2219
  - std::discard\_block\_engine, 2402
  - std::discrete\_distribution, 2407
  - std::exponential\_distribution, 2422
  - std::extreme\_value\_distribution, 2426
  - std::fisher\_f\_distribution, 2430
  - std::gamma\_distribution, 2481
  - std::geometric\_distribution, 2486
  - std::independent\_bits\_engine, 2538
  - std::linear\_congruential\_engine, 2602
  - std::lognormal\_distribution, 2653
  - std::negative\_binomial\_distribution, 2791
  - std::normal\_distribution, 2797
  - std::numeric\_limits, 2840
  - std::piecewise\_constant\_distribution, 2903
  - std::piecewise\_linear\_distribution, 2909
  - std::poisson\_distribution, 2919
  - std::shuffle\_order\_engine, 3016
  - std::student\_t\_distribution, 3028
  - std::uniform\_int\_distribution, 3124
  - std::uniform\_real\_distribution, 3128
  - std::weibull\_distribution, 3178
- max\_digits10
  - std::\_\_numeric\_limits\_base, 1397
  - std::numeric\_limits, 2844
- max\_element
  - sorting\_algorithms, 172
- max\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1208
- max\_exponent
  - std::\_\_numeric\_limits\_base, 1397
  - std::numeric\_limits, 2844
- max\_exponent10
  - std::\_\_numeric\_limits\_base, 1398
  - std::numeric\_limits, 2844
- max\_size
  - \_\_gnu\_cxx::\_\_versa\_string, 830
  - \_\_gnu\_debug::basic\_string, 1076
  - std::basic\_string, 2069
  - std::deque, 2392
  - std::forward\_list, 2449
  - std::list, 2621
  - std::map, 2672
  - std::match\_results, 2688
  - std::multimap, 2764
  - std::multiset, 2783
  - std::set, 2993
  - std::vector, 3166
- mean
  - std::normal\_distribution, 2797
  - std::poisson\_distribution, 2919
- mem\_fn
  - functors, 253
- Memory, 65
- memory, 3393
- memory\_order
  - atomics, 209
- merge
  - sorting\_algorithms, 173
  - std::forward\_list, 2449
  - std::list, 2621, 2622
- merge.h, 3394
- merge\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1209
- merge\_oversampling
  - \_\_gnu\_parallel::\_Settings, 1209
- messages
  - std::locale, 2638
  - std::messages, 2701
- messages\_members.h, 3395
- metaprogramming
  - false\_type, 121
  - true\_type, 121
- microseconds
  - std::chrono, 729



- 
- milliseconds
    - std::chrono, [729](#)
  - min
    - \_\_gnu\_parallel, [438](#)
    - numeric\_arrays, [91](#)
    - sorting\_algorithms, [174](#)
    - std::bernoulli\_distribution, [2180](#)
    - std::binomial\_distribution, [2191](#)
    - std::cauchy\_distribution, [2210](#)
    - std::chi\_squared\_distribution, [2219](#)
    - std::discard\_block\_engine, [2402](#)
    - std::discrete\_distribution, [2407](#)
    - std::exponential\_distribution, [2422](#)
    - std::extreme\_value\_distribution, [2426](#)
    - std::fisher\_f\_distribution, [2430](#)
    - std::gamma\_distribution, [2481](#)
    - std::geometric\_distribution, [2486](#)
    - std::independent\_bits\_engine, [2538](#)
    - std::linear\_congruential\_engine, [2602](#)
    - std::lognormal\_distribution, [2653](#)
    - std::negative\_binomial\_distribution, [2791](#)
    - std::normal\_distribution, [2797](#)
    - std::numeric\_limits, [2841](#)
    - std::piecewise\_constant\_distribution, [2903](#)
    - std::piecewise\_linear\_distribution, [2909](#)
    - std::poisson\_distribution, [2919](#)
    - std::shuffle\_order\_engine, [3016](#)
    - std::student\_t\_distribution, [3028](#)
    - std::uniform\_int\_distribution, [3124](#)
    - std::uniform\_real\_distribution, [3128](#)
    - std::weibull\_distribution, [3178](#)
  - min\_element
    - sorting\_algorithms, [175](#)
  - min\_element\_minimal\_n
    - \_\_gnu\_parallel::Settings, [1209](#)
  - min\_exponent
    - std::\_\_numeric\_limits\_base, [1398](#)
    - std::numeric\_limits, [2844](#)
  - min\_exponent10
    - std::\_\_numeric\_limits\_base, [1398](#)
    - std::numeric\_limits, [2844](#)
  - minmax
    - sorting\_algorithms, [176](#)
  - minmax\_element
    - sorting\_algorithms, [176](#), [177](#)
  - minstd\_rand
    - random\_generators, [279](#)
  - minstd\_rand0
    - random\_generators, [279](#)
  - minutes
    - std::chrono, [729](#)
  - mismatch
    - non\_mutating\_algorithms, [159](#)
  - modulus
    - std::linear\_congruential\_engine, [2605](#)
  - monetary
    - std::locale, [2638](#)
  - money\_get
    - std::money\_get, [2715](#)
  - money\_put
    - std::money\_put, [2720](#)
  - money\_punct
    - std::money\_punct, [2727](#)
  - move
    - mutating\_algorithms, [134](#)
  - move.h, [3395](#)
  - move\_backward
    - mutating\_algorithms, [134](#)
  - mt19937
    - random\_generators, [279](#)
  - mt19937\_64
    - random\_generators, [279](#)
  - mt\_allocator.h, [3397](#)
  - multimap
    - std::multimap, [2752–2754](#)
  - multimap.h, [3398](#), [3399](#)
  - multiplier
    - std::linear\_congruential\_engine, [2605](#)
  - multiseq\_partition
    - \_\_gnu\_parallel, [438](#)
  - multiseq\_selection
    - \_\_gnu\_parallel, [439](#)
  - multiseq\_selection.h, [3400](#)
  - multiset
    - std::multiset, [2772–2774](#)
-

- multiset.h, 3401, 3402
- multiway\_merge
  - \_\_gnu\_parallel, 440
- multiway\_merge.h, 3403
  - \_GLIBCXX\_PARALLEL\_-LENGTH, 3409
- multiway\_merge\_3\_variant
  - \_\_gnu\_parallel, 442
- multiway\_merge\_4\_variant
  - \_\_gnu\_parallel, 442
- multiway\_merge\_exact\_splitting
  - \_\_gnu\_parallel, 443
- multiway\_merge\_loser\_tree
  - \_\_gnu\_parallel, 444
- multiway\_merge\_loser\_tree\_sentinel
  - \_\_gnu\_parallel, 444
- multiway\_merge\_loser\_tree\_unguarded
  - \_\_gnu\_parallel, 445
- multiway\_merge\_minimal\_k
  - \_\_gnu\_parallel::\_Settings, 1209
- multiway\_merge\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1209
- multiway\_merge\_oversampling
  - \_\_gnu\_parallel::\_Settings, 1209
- multiway\_merge\_sampling\_splitting
  - \_\_gnu\_parallel, 446
- multiway\_merge\_sentinels
  - \_\_gnu\_parallel, 446
- multiway\_mergesort.h, 3409
- Mutating, 126
- mutating\_algorithms
  - copy, 129
  - copy\_backward, 129
  - copy\_if, 130
  - copy\_n, 130
  - fill, 131
  - fill\_n, 131
  - generate, 132
  - generate\_n, 132
  - is\_partitioned, 133
  - iter\_swap, 133
  - move, 134
  - move\_backward, 134
  - partition, 135
  - partition\_copy, 135
  - partition\_point, 136
  - random\_shuffle, 136, 137
  - remove, 137
  - remove\_copy, 138
  - remove\_copy\_if, 138
  - remove\_if, 139
  - replace, 140
  - replace\_copy\_if, 140
  - replace\_if, 141
  - reverse, 141
  - reverse\_copy, 142
  - rotate, 142
  - rotate\_copy, 143
  - shuffle, 143
  - stable\_partition, 144
  - swap, 145
  - swap\_ranges, 145
  - transform, 145, 146
  - unique, 147
  - unique\_copy, 148
- mutex, 3410
- Mutexes, 69
- mutexes
  - call\_once, 71
  - lock, 71
  - try\_lock, 71
- name
  - std::locale, 2635
  - std::type\_info, 3116
- nanoseconds
  - std::chrono, 729
- narrow
  - std::\_\_ctype\_abstract\_base, 1319
  - std::basic\_fstream, 1569
  - std::basic\_ifstream, 1632
  - std::basic\_ios, 1675
  - std::basic\_iostream, 1718
  - std::basic\_istream, 1778
  - std::basic\_istreamstream, 1833
  - std::basic\_ofstream, 1880
  - std::basic\_ostream, 1923
  - std::basic\_ostreamstream, 1968
  - std::basic\_stringstream, 2143
  - std::ctype, 2291, 2292
  - std::ctype< char >, 2305, 2306
  - std::ctype< wchar\_t >, 2327

- std::ctype\_byname, [2342](#), [2343](#)
- std::ctype\_byname< char >, [2355](#), [2356](#)
- native\_handle
  - std::thread, [3039](#)
- neg\_format
  - std::moneypunct, [2733](#)
  - std::moneypunct\_byname, [2745](#)
- negative\_sign
  - std::moneypunct, [2734](#)
  - std::moneypunct\_byname, [2746](#)
- Negators, [256](#)
- negators
  - not1, [257](#)
  - not2, [257](#)
- nested\_exception.h, [3412](#)
- new, [3413](#)
  - operator delete, [3414](#)
  - operator new, [3416](#)
- new\_allocator.h, [3418](#)
- new\_handler
  - std, [599](#)
- next\_permutation
  - sorting\_algorithms, [177](#), [178](#)
- noboolalpha
  - std, [628](#)
- Non-Mutating, [149](#)
- non\_mutating\_algorithms
  - adjacent\_find, [151](#)
  - all\_of, [152](#)
  - any\_of, [152](#)
  - count, [153](#)
  - count\_if, [153](#)
  - equal, [153](#), [154](#)
  - find, [154](#)
  - find\_end, [155](#), [156](#)
  - find\_first\_of, [156](#), [157](#)
  - find\_if, [158](#)
  - find\_if\_not, [158](#)
  - for\_each, [158](#)
  - mismatch, [159](#)
  - none\_of, [160](#)
  - search, [160](#), [161](#)
  - search\_n, [162](#)
- none
  - std::bitset, [2202](#)
  - std::locale, [2639](#)
- none\_of
  - non\_mutating\_algorithms, [160](#)
- norm
  - complex\_numbers, [46](#)
- Normal, [288](#)
- normal\_distribution
  - std::normal\_distribution, [2797](#)
- noshowbase
  - std, [629](#)
- noshowpoint
  - std, [629](#)
- noshowpos
  - std, [629](#)
- noskipws
  - std, [629](#)
- nosubs
  - std::regex\_constants, [750](#)
- not1
  - negators, [257](#)
- not2
  - negators, [257](#)
- nounitbuf
  - std, [629](#)
- nouppercase
  - std, [629](#)
- npos
  - \_\_gnu\_cxx::\_\_versa\_string, [848](#)
  - \_\_gnu\_debug::basic\_string, [1091](#)
  - std::basic\_string, [2086](#)
- nth\_element
  - sorting\_algorithms, [178](#), [179](#)
- nth\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [1210](#)
- num\_get
  - std::num\_get, [2807](#)
- num\_put
  - std::num\_put, [2827](#)
- numeric, [3419](#), [3420](#)
  - std::locale, [2639](#)
- Numeric Arrays, [77](#)
- numeric\_arrays
  - ~gslice, [90](#)
  - apply, [90](#)
  - cshift, [91](#)
  - gslice, [87](#)

- gslice\_array, 87
- indirect\_array, 87
- mask\_array, 88
- max, 91
- min, 91
- operator<=, 95
- operator>=, 100
- operator\*=, 93
- operator~, 105
- operator^=, 104, 105
- operator+, 93
- operator+=, 93, 94
- operator-, 94
- operator=, 94
- operator/=, 95
- operator=, 96–100
- operator%=, 92
- operator&=, 92
- resize, 106
- shift, 106
- size, 106, 107
- slice, 88
- slice\_array, 88
- start, 107
- stride, 107
- sum, 107
- valarray, 88–90
- numeric\_traits.h, 3423
- numeric\_fwd.h, 3423
- Numerics, 72
- numpunct
  - std::numpunct, 2872, 2873
- oct
  - std, 630
  - std::basic\_fstream, 1603
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1687
  - std::basic\_iostream, 1752
  - std::basic\_istream, 1803
  - std::basic\_istreamstream, 1858
  - std::basic\_ofstream, 1902
  - std::basic\_ostream, 1945
  - std::basic\_ostreamstream, 1990
  - std::basic\_stringstream, 2177
  - std::ios\_base, 2567
- off\_type
  - \_\_gnu\_cxx::enc\_filebuf, 877
  - \_\_gnu\_cxx::stdio\_filebuf, 946
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 978
  - std::basic\_filebuf, 1513
  - std::basic\_fstream, 1551
  - std::basic\_ifstream, 1615
  - std::basic\_ios, 1666
  - std::basic\_iostream, 1701
  - std::basic\_istream, 1763
  - std::basic\_istreamstream, 1817
  - std::basic\_ofstream, 1869
  - std::basic\_ostream, 1913
  - std::basic\_ostreamstream, 1957
  - std::basic\_streambuf, 2008
  - std::basic\_stringbuf, 2091
  - std::basic\_stringstream, 2124, 2125
- ofstream
  - io, 61
- omp\_loop.h, 3426
- omp\_loop\_static.h, 3427
- open
  - \_\_gnu\_cxx::enc\_filebuf, 882
  - \_\_gnu\_cxx::stdio\_filebuf, 953, 954
  - std::basic\_filebuf, 1519
  - std::basic\_fstream, 1570
  - std::basic\_ifstream, 1632
  - std::basic\_ofstream, 1881
- openmode
  - std::basic\_fstream, 1551
  - std::basic\_ifstream, 1615
  - std::basic\_ios, 1666
  - std::basic\_iostream, 1701
  - std::basic\_istream, 1763
  - std::basic\_istreamstream, 1817
  - std::basic\_ofstream, 1870
  - std::basic\_ostream, 1913
  - std::basic\_ostreamstream, 1958
  - std::basic\_stringstream, 2125
  - std::ios\_base, 2556
- operator\_iterator
  - \_\_gnu\_debug::\_Safe\_iterator, 1023
- operator\_RAlter
  - \_\_gnu\_parallel::\_GuardedIterator, 1146
- operator bool

- std::basic\_istream::sentry, 1807
- std::basic\_ostream::sentry, 1948
- std::function< \_Res(\_- ArgTypes...)>, 2466
- operator delete
  - new, 3414
- operator new
  - new, 3416
- operator streamoff
  - std::fpos, 2457
- operator string\_type
  - std::sub\_match, 3035
- operator void \*
  - std::basic\_fstream, 1570
  - std::basic\_ifstream, 1633
  - std::basic\_ios, 1675
  - std::basic\_iostream, 1719
  - std::basic\_istream, 1779
  - std::basic\_istream, 1834
  - std::basic\_ofstream, 1881
  - std::basic\_ostream, 1923
  - std::basic\_ostringstream, 1968
  - std::basic\_stringstream, 2143
- operator<
  - \_\_gnu\_cxx, 365, 366
  - \_\_gnu\_parallel::\_GuardedIterator, 1147
  - regex, 225–228
  - std, 636–642
  - std::multiset, 2786
- operator<<
  - complex\_numbers, 50
  - pointer\_abstractions, 69
  - random\_distributions\_bernoulli, 294
  - random\_distributions\_normal, 290
  - random\_distributions\_poisson, 300, 301
  - random\_distributions\_uniform, 285, 286
  - random\_generators, 282
  - regex, 228
  - std, 642–648
  - std::basic\_fstream, 1571–1577
  - std::basic\_iostream, 1719–1726
  - std::basic\_ofstream, 1882–1888
  - std::basic\_ostream, 1924–1930
- std::basic\_ostringstream, 1969–1975
- std::basic\_stringstream, 2143–2150
- std::binomial\_distribution, 2193
- std::bitset, 2203
- std::chi\_squared\_distribution, 2220
- std::discard\_block\_engine, 2404
- std::discrete\_distribution, 2408
- std::fisher\_f\_distribution, 2431
- std::gamma\_distribution, 2483
- std::linear\_congruential\_engine, 2604
- std::lognormal\_distribution, 2654
- std::negative\_binomial\_distribution, 2793
- std::normal\_distribution, 2799
- std::piecewise\_constant\_distribution, 2905
- std::piecewise\_linear\_distribution, 2910
- std::poisson\_distribution, 2921
- std::shuffle\_order\_engine, 3018
- std::student\_t\_distribution, 3029
- operator<=<=
  - numeric\_arrays, 95
  - std::bitset, 2203
  - std::indirect\_array, 2543
  - std::mask\_array, 2679
  - std::slice\_array, 3022
- operator<=
  - \_\_gnu\_cxx, 367, 368
  - \_\_gnu\_parallel::\_GuardedIterator, 1148
  - regex, 228–231
  - std, 649–652
  - std::rel\_ops, 752
- operator>
  - \_\_gnu\_cxx, 370
  - regex, 234–237
  - std, 659–662
  - std::rel\_ops, 752
- operator>>
  - complex\_numbers, 52
  - random\_distributions\_bernoulli, 295, 296
  - random\_distributions\_normal, 291

- random\_distributions\_poisson, [303](#), [304](#)
- random\_distributions\_uniform, [287](#)
- std, [666–672](#)
- std::basic\_fstream, [1578–1584](#)
- std::basic\_ifstream, [1633–1640](#)
- std::basic\_iostream, [1726–1732](#)
- std::basic\_istream, [1779–1786](#)
- std::basic\_istream, [1834–1840](#)
- std::basic\_stringstream, [2150–2156](#)
- std::binomial\_distribution, [2193](#)
- std::bitset, [2203](#)
- std::chi\_squared\_distribution, [2220](#)
- std::discard\_block\_engine, [2404](#)
- std::discrete\_distribution, [2408](#)
- std::fisher\_f\_distribution, [2431](#)
- std::gamma\_distribution, [2484](#)
- std::independent\_bits\_engine, [2540](#)
- std::linear\_congruential\_engine, [2604](#)
- std::lognormal\_distribution, [2655](#)
- std::negative\_binomial\_distribution, [2793](#)
- std::normal\_distribution, [2800](#)
- std::piecewise\_constant\_distribution, [2905](#)
- std::piecewise\_linear\_distribution, [2910](#)
- std::poisson\_distribution, [2921](#)
- std::shuffle\_order\_engine, [3018](#)
- std::student\_t\_distribution, [3030](#)
- operator>>=
  - numeric\_arrays, [100](#)
  - std::bitset, [2204](#)
  - std::indirect\_array, [2543](#)
  - std::mask\_array, [2679](#)
  - std::slice\_array, [3022](#)
- operator>=
  - \_\_gnu\_cxx, [371](#), [372](#)
  - regex, [237–239](#)
  - std, [662–666](#)
  - std::rel\_ops, [752](#)
- operator\*
  - \_\_gnu\_debug::Safe\_iterator, [1023](#)
  - \_\_gnu\_parallel::\_GuardedIterator, [1147](#)
  - complex\_numbers, [47](#)
  - std::auto\_ptr, [1496](#)
  - std::back\_insert\_iterator, [1501](#)
  - std::front\_insert\_iterator, [2461](#)
  - std::insert\_iterator, [2548](#)
  - std::istreambuf\_iterator, [2588](#)
  - std::ostreambuf\_iterator, [2893](#)
  - std::regex\_iterator, [2952](#)
  - std::regex\_token\_iterator, [2958](#)
  - std::reverse\_iterator, [2970](#)
- operator\*=
  - complex\_numbers, [48](#)
  - numeric\_arrays, [93](#)
  - std::indirect\_array, [2543](#)
  - std::mask\_array, [2679](#)
  - std::slice\_array, [3022](#)
- operator~
  - numeric\_arrays, [105](#)
  - std::bitset, [2206](#)
- operator^
  - std, [673](#)
- operator^=
  - numeric\_arrays, [104](#), [105](#)
  - std::bitset, [2205](#)
  - std::indirect\_array, [2543](#)
  - std::mask\_array, [2679](#)
  - std::slice\_array, [3022](#)
- operator()
  - \_\_gnu\_cxx::subtractive\_rng, [999](#)
  - \_\_gnu\_parallel::\_Nothing, [1188](#)
  - \_\_gnu\_parallel::\_RandomNumber, [1200](#)
  - \_\_gnu\_parallel::\_accumulate\_selector, [1093](#)
  - \_\_gnu\_parallel::\_adjacent\_find\_selector, [1096](#)
  - \_\_gnu\_parallel::\_count\_if\_selector, [1101](#)
  - \_\_gnu\_parallel::\_count\_selector, [1103](#)
  - \_\_gnu\_parallel::\_fill\_selector, [1105](#)
  - \_\_gnu\_parallel::\_find\_first\_of\_selector, [1107](#)
  - \_\_gnu\_parallel::\_find\_if\_selector, [1109](#)

- 
- \_\_gnu\_parallel::\_\_for\_each\_-  
  selector, 1110
  - \_\_gnu\_parallel::\_\_generate\_-  
  selector, 1112
  - \_\_gnu\_parallel::\_\_identity\_selector,  
  1116
  - \_\_gnu\_parallel::\_\_inner\_product\_-  
  selector, 1118
  - \_\_gnu\_parallel::\_\_mismatch\_-  
  selector, 1122
  - \_\_gnu\_parallel::\_\_replace\_if\_-  
  selector, 1128
  - \_\_gnu\_parallel::\_\_replace\_selector,  
  1130
  - \_\_gnu\_parallel::\_\_transform1\_-  
  selector, 1132
  - \_\_gnu\_parallel::\_\_transform2\_-  
  selector, 1134
  - std::bernoulli\_distribution, 2180
  - std::binomial\_distribution, 2191
  - std::cauchy\_distribution, 2210
  - std::chi\_squared\_distribution, 2219
  - std::discard\_block\_engine, 2402
  - std::discrete\_distribution, 2407
  - std::exponential\_distribution, 2422
  - std::extreme\_value\_distribution,  
  2426
  - std::fisher\_f\_distribution, 2430
  - std::function<\_Res(\_-  
  ArgTypes...)>, 2466
  - std::gamma\_distribution, 2482
  - std::geometric\_distribution, 2487
  - std::independent\_bits\_engine, 2539
  - std::linear\_congruential\_engine,  
  2602
  - std::locale, 2636
  - std::lognormal\_distribution, 2653
  - std::negative\_binomial\_distribution,  
  2791
  - std::normal\_distribution, 2798
  - std::piecewise\_constant\_-  
  distribution, 2904
  - std::piecewise\_linear\_distribution,  
  2909
  - std::poisson\_distribution, 2919
  - std::shuffle\_order\_engine, 3016
  - std::student\_t\_distribution, 3028
  - std::uniform\_int\_distribution, 3125
  - std::uniform\_real\_distribution, 3128
  - std::weibull\_distribution, 3179
  - operator+
    - \_\_gnu\_cxx, 363–365
    - complex\_numbers, 48
    - numeric\_arrays, 93
    - std, 634–636
    - std::fpos, 2457
    - std::reverse\_iterator, 2971
  - operator++
    - \_\_gnu\_debug::\_Safe\_iterator, 1023
    - \_\_gnu\_parallel::\_GuardedIterator,  
  1147
    - std::back\_insert\_iterator, 1501, 1502
    - std::front\_insert\_iterator, 2461
    - std::insert\_iterator, 2548, 2549
    - std::istreambuf\_iterator, 2588
    - std::ostreambuf\_iterator, 2893
    - std::regex\_iterator, 2952
    - std::regex\_token\_iterator, 2958
    - std::reverse\_iterator, 2971
  - operator+=
    - \_\_gnu\_cxx::\_\_versa\_string, 830–  
  832
    - \_\_gnu\_debug::basic\_string, 1076,  
  1077
    - complex\_numbers, 49
    - numeric\_arrays, 93, 94
    - std::basic\_string, 2069, 2070
    - std::complex, 2270
    - std::fpos, 2457
    - std::indirect\_array, 2543
    - std::mask\_array, 2679
    - std::reverse\_iterator, 2971
    - std::slice\_array, 3022
  - operator-
    - complex\_numbers, 49
    - numeric\_arrays, 94
    - std::fpos, 2457
    - std::reverse\_iterator, 2972
  - operator->
    - \_\_gnu\_debug::\_Safe\_iterator, 1024
    - std::auto\_ptr, 1496
    - std::regex\_iterator, 2953
-

- std::regex\_token\_iterator, 2959
- std::reverse\_iterator, 2973
- operator--
  - \_\_gnu\_debug::\_Safe\_iterator, 1024
  - std::reverse\_iterator, 2972
- operator==
  - complex\_numbers, 49
  - numeric\_arrays, 94
  - std::complex, 2270
  - std::fpos, 2458
  - std::indirect\_array, 2543
  - std::mask\_array, 2679
  - std::reverse\_iterator, 2973
  - std::slice\_array, 3022
- operator/
  - complex\_numbers, 50
- operator/=
  - complex\_numbers, 50
  - numeric\_arrays, 95
  - std::indirect\_array, 2543
  - std::mask\_array, 2679
  - std::slice\_array, 3022
- operator=
  - \_\_gnu\_cxx::\_\_versa\_string, 832, 833
  - \_\_gnu\_debug::\_Safe\_iterator, 1025
  - complex\_numbers, 51
  - numeric\_arrays, 96–100
  - std::auto\_ptr, 1496, 1497
  - std::back\_insert\_iterator, 1502
  - std::basic\_regex, 2001, 2002
  - std::basic\_string, 2071, 2072
  - std::deque, 2392, 2393
  - std::forward\_list, 2450
  - std::front\_insert\_iterator, 2462
  - std::function<\_Res(\_- ArgTypes...)>, 2467–2469
  - std::insert\_iterator, 2549
  - std::list, 2622, 2623
  - std::locale, 2636
  - std::map, 2672, 2673
  - std::match\_results, 2688
  - std::multimap, 2764, 2765
  - std::multiset, 2783, 2784
  - std::ostream\_iterator, 2889
  - std::ostreambuf\_iterator, 2893
  - std::regex\_iterator, 2953
  - std::regex\_token\_iterator, 2959
  - std::set, 2993, 2994
  - std::vector, 3166, 3167
- operator==
  - \_\_gnu\_cxx, 368, 369
  - complex\_numbers, 51
  - iterators, 272, 273
  - random\_distributions\_bernoulli, 295
  - random\_distributions\_normal, 291
  - random\_distributions\_poisson, 301, 302
  - random\_distributions\_uniform, 286
  - regex, 231–234
  - std, 652–659
  - std::binomial\_distribution, 2193
  - std::bitset, 2203
  - std::chi\_squared\_distribution, 2220
  - std::discard\_block\_engine, 2404
  - std::fisher\_f\_distribution, 2431
  - std::gamma\_distribution, 2483
  - std::independent\_bits\_engine, 2540
  - std::linear\_congruential\_engine, 2604
  - std::locale, 2636
  - std::lognormal\_distribution, 2654
  - std::multiset, 2786
  - std::negative\_binomial\_distribution, 2793
  - std::normal\_distribution, 2800
  - std::poisson\_distribution, 2921
  - std::regex\_iterator, 2953
  - std::regex\_token\_iterator, 2959
  - std::shuffle\_order\_engine, 3018
  - std::student\_t\_distribution, 3029
- operator%=
  - numeric\_arrays, 92
  - std::indirect\_array, 2542
  - std::mask\_array, 2678
  - std::slice\_array, 3021
- operator&
  - std, 634
- operator&=
  - numeric\_arrays, 92
  - std::bitset, 2203
  - std::indirect\_array, 2542



- 
- std::mask\_array, [2678](#)
  - std::slice\_array, [3021](#)
  - optimize
    - std::regex\_constants, [750](#)
  - os\_defines.h, [3427](#)
  - ostream, [3428](#)
    - io, [62](#)
  - ostream.tcc, [3429](#)
  - ostream\_insert.h, [3430](#)
  - ostream\_iterator
    - std::ostream\_iterator, [2888](#)
  - ostream\_type
    - std::ostream\_iterator, [2887](#)
    - std::ostreambuf\_iterator, [2891](#)
  - ostreambuf\_iterator
    - std::ostreambuf\_iterator, [2892](#)
  - ostringstream
    - io, [62](#)
  - out
    - std::\_\_codecvt\_abstract\_base, [1306](#)
    - std::basic\_fstream, [1604](#)
    - std::basic\_ifstream, [1657](#)
    - std::basic\_ios, [1688](#)
    - std::basic\_istream, [1752](#)
    - std::basic\_istream, [1803](#)
    - std::basic\_istream, [1859](#)
    - std::basic\_ofstream, [1903](#)
    - std::basic\_ostream, [1945](#)
    - std::basic\_ostringstream, [1990](#)
    - std::basic\_stringstream, [2177](#)
    - std::codecvt, [2230](#)
    - std::codecvt< \_InternT, \_ExternT, encoding\_state >, [2236](#)
    - std::codecvt< char, char, mbstate\_t >, [2241](#)
    - std::codecvt< wchar\_t, char, mbstate\_t >, [2246](#)
    - std::codecvt\_byname, [2253](#)
    - std::ios\_base, [2568](#)
  - overflow
    - \_\_gnu\_cxx::enc\_filebuf, [883](#)
    - \_\_gnu\_cxx::stdio\_filebuf, [954](#)
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, [982](#)
    - std::basic\_filebuf, [1520](#)
    - std::basic\_streambuf, [2013](#)
    - std::basic\_stringbuf, [2095](#)
  - p
    - std::bernoulli\_distribution, [2180](#)
    - std::binomial\_distribution, [2192](#)
    - std::geometric\_distribution, [2487](#)
    - std::negative\_binomial\_distribution, [2791](#)
  - pair
    - std::pair, [2900](#), [2901](#)
  - par\_loop.h, [3430](#)
  - parallel.h, [3431](#)
  - parallel\_balanced
    - \_\_gnu\_parallel, [403](#)
  - parallel\_omp\_loop
    - \_\_gnu\_parallel, [403](#)
  - parallel\_omp\_loop\_static
    - \_\_gnu\_parallel, [403](#)
  - parallel\_taskqueue
    - \_\_gnu\_parallel, [403](#)
  - parallel\_unbalanced
    - \_\_gnu\_parallel, [403](#)
  - parallel\_multiway\_merge
    - \_\_gnu\_parallel, [448](#)
  - parallel\_sort\_mwms
    - \_\_gnu\_parallel, [449](#)
  - parallel\_sort\_mwms\_pu
    - \_\_gnu\_parallel, [449](#)
  - parallel\_tag
    - \_\_gnu\_parallel::parallel\_tag, [1235](#)
  - param
    - std::bernoulli\_distribution, [2180](#), [2181](#)
    - std::binomial\_distribution, [2192](#)
    - std::cauchy\_distribution, [2211](#)
    - std::chi\_squared\_distribution, [2219](#)
    - std::discrete\_distribution, [2407](#)
    - std::exponential\_distribution, [2423](#)
    - std::extreme\_value\_distribution, [2427](#)
    - std::fisher\_f\_distribution, [2430](#)
    - std::gamma\_distribution, [2482](#)
    - std::geometric\_distribution, [2487](#)
    - std::lognormal\_distribution, [2653](#), [2654](#)
    - std::negative\_binomial\_distribution, [2792](#)
    - std::normal\_distribution, [2798](#)
-

- std::piecewise\_constant\_distribution, [2904](#)
- std::piecewise\_linear\_distribution, [2909](#)
- std::poisson\_distribution, [2920](#)
- std::student\_t\_distribution, [3028](#)
- std::uniform\_int\_distribution, [3125](#)
- std::uniform\_real\_distribution, [3129](#)
- std::weibull\_distribution, [3179](#)
- partial\_sort
  - sorting\_algorithms, [180](#)
- partial\_sort\_copy
  - sorting\_algorithms, [181](#)
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [1210](#)
- partial\_sum
  - std, [673](#), [674](#)
- partial\_sum.h, [3431](#)
- partial\_sum\_dilation
  - \_\_gnu\_parallel::\_Settings, [1210](#)
- partial\_sum\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [1210](#)
- partition
  - mutating\_algorithms, [135](#)
- partition.h, [3432](#)
  - \_GLIBCXX\_VOLATILE, [3433](#)
- partition\_chunk\_share
  - \_\_gnu\_parallel::\_Settings, [1210](#)
- partition\_chunk\_size
  - \_\_gnu\_parallel::\_Settings, [1210](#)
- partition\_copy
  - mutating\_algorithms, [135](#)
- partition\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [1211](#)
- partition\_point
  - mutating\_algorithms, [136](#)
- pbackfail
  - \_\_gnu\_cxx::enc\_filebuf, [883](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [955](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [983](#)
  - std::basic\_filebuf, [1521](#)
  - std::basic\_streambuf, [2013](#)
  - std::basic\_stringbuf, [2096](#)
- pbase
  - \_\_gnu\_cxx::enc\_filebuf, [884](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [955](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [983](#)
  - std::basic\_filebuf, [1521](#)
  - std::basic\_streambuf, [2014](#)
  - std::basic\_stringbuf, [2097](#)
- pbump
  - \_\_gnu\_cxx::enc\_filebuf, [884](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [956](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [984](#)
  - std::basic\_filebuf, [1522](#)
  - std::basic\_streambuf, [2014](#)
  - std::basic\_stringbuf, [2097](#)
- peek
  - std::basic\_fstream, [1584](#)
  - std::basic\_ifstream, [1640](#)
  - std::basic\_iostream, [1733](#)
  - std::basic\_istream, [1786](#)
  - std::basic\_istreamstream, [1841](#)
  - std::basic\_stringstream, [2157](#)
- pod\_char\_traits.h, [3433](#)
- pointer
  - std::back\_insert\_iterator, [1500](#)
  - std::front\_insert\_iterator, [2460](#)
  - std::insert\_iterator, [2547](#)
  - std::istream\_iterator, [2583](#)
  - std::istreambuf\_iterator, [2586](#)
  - std::iterator, [2590](#)
  - std::ostream\_iterator, [2887](#)
  - std::ostreambuf\_iterator, [2891](#)
  - std::raw\_storage\_iterator, [2944](#)
  - std::reverse\_iterator, [2969](#)
  - std::set, [2981](#)
- Pointer Abstractions, [65](#)
- pointer.h, [3434](#)
- pointer\_abstractions
  - allocate\_shared, [68](#)
  - get\_deleter, [68](#)
  - make\_shared, [68](#)
  - operator<<, [69](#)
- pointer\_adaptors
  - ptr\_fun, [258](#)
- Poisson, [296](#)
- polar
  - complex\_numbers, [52](#)
- Policy-Based Data Structures, [275](#)
- pool\_allocator.h, [3437](#)
- pop

- std::priority\_queue, 2925
- std::queue, 2932
- std::stack, 3025
- pop\_back
  - \_\_gnu\_parallel::-  
    RestrictedBoundedConcurrentQueue, 1202
  - std::deque, 2394
  - std::list, 2623
  - std::vector, 3168
- pop\_front
  - \_\_gnu\_parallel::-  
    RestrictedBoundedConcurrentQueue, 1202
  - std::deque, 2395
  - std::forward\_list, 2451
  - std::list, 2623
- pop\_heap
  - heap\_algorithms, 264
- pos\_format
  - std::moneypunct, 2734
  - std::moneypunct\_byname, 2746
- pos\_type
  - \_\_gnu\_cxx::enc\_filebuf, 877
  - \_\_gnu\_cxx::stdio\_filebuf, 946
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 978
  - std::basic\_filebuf, 1513
  - std::basic\_fstream, 1552
  - std::basic\_ifstream, 1615
  - std::basic\_ios, 1666
  - std::basic\_iostream, 1702
  - std::basic\_istream, 1764
  - std::basic\_istream, 1817
  - std::basic\_ofstream, 1870
  - std::basic\_ostream, 1914
  - std::basic\_ostringstream, 1958
  - std::basic\_streambuf, 2008
  - std::basic\_stringbuf, 2091
  - std::basic\_stringstream, 2125, 2126
- position
  - std::match\_results, 2688
- positive\_sign
  - std::moneypunct, 2735
  - std::moneypunct\_byname, 2747
- postypes.h, 3437
- pow
  - complex\_numbers, 52
- power
  - SGIextensions, 23
- pptr
  - \_\_gnu\_cxx::enc\_filebuf, 885
  - \_\_gnu\_cxx::stdio\_filebuf, 956
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 984
  - std::basic\_filebuf, 1522
  - std::basic\_streambuf, 2014
  - std::basic\_stringbuf, 2097
- precision
  - std::basic\_fstream, 1584, 1585
  - std::basic\_ifstream, 1640, 1641
  - std::basic\_ios, 1676
  - std::basic\_iostream, 1733
  - std::basic\_istream, 1786, 1787
  - std::basic\_istream, 1841, 1842
  - std::basic\_ofstream, 1889
  - std::basic\_ostream, 1930, 1931
  - std::basic\_ostringstream, 1975, 1976
  - std::basic\_stringstream, 2157
  - std::ios\_base, 2559, 2560
- prefix
  - std::match\_results, 2689
- prev\_permutation
  - sorting\_algorithms, 182, 183
- priority\_queue
  - std::priority\_queue, 2924
- priority\_queue.hpp, 3438
- priority\_queue\_base\_dispatch.hpp, 3439
- probabilities
  - std::discrete\_distribution, 2408
- profiler.h, 3439
- profiler\_algos.h, 3442
- profiler\_hashtable\_size.h, 3443
- profiler\_list\_to\_slist.h, 3444
- profiler\_list\_to\_vector.h, 3444
- profiler\_map\_to\_unordered\_map.h, 3445
- profiler\_node.h, 3447
- profiler\_state.h, 3448
- profiler\_trace.h, 3448
- profiler\_vector\_size.h, 3451
- profiler\_vector\_to\_list.h, 3451
- ptr\_fun
  - pointer\_adaptors, 258
- pubimbue

- `__gnu_cxx::enc_filebuf`, 885
  - `__gnu_cxx::stdio_filebuf`, 957
  - `__gnu_cxx::stdio_sync_filebuf`, 985
  - `std::basic_filebuf`, 1522
  - `std::basic_streambuf`, 2015
  - `std::basic_stringbuf`, 2098
- pubseekoff
  - `__gnu_cxx::enc_filebuf`, 885
  - `__gnu_cxx::stdio_filebuf`, 957
  - `__gnu_cxx::stdio_sync_filebuf`, 985
  - `std::basic_filebuf`, 1523
  - `std::basic_streambuf`, 2015
  - `std::basic_stringbuf`, 2098
- pubseekpos
  - `__gnu_cxx::enc_filebuf`, 886
  - `__gnu_cxx::stdio_filebuf`, 957
  - `__gnu_cxx::stdio_sync_filebuf`, 985
  - `std::basic_filebuf`, 1523
  - `std::basic_streambuf`, 2016
  - `std::basic_stringbuf`, 2098
- pubsetbuf
  - `__gnu_cxx::enc_filebuf`, 886
  - `__gnu_cxx::stdio_filebuf`, 958
  - `__gnu_cxx::stdio_sync_filebuf`, 986
  - `std::basic_filebuf`, 1523
  - `std::basic_streambuf`, 2016
  - `std::basic_stringbuf`, 2099
- pubsync
  - `__gnu_cxx::enc_filebuf`, 886
  - `__gnu_cxx::stdio_filebuf`, 958
  - `__gnu_cxx::stdio_sync_filebuf`, 986
  - `std::basic_filebuf`, 1524
  - `std::basic_streambuf`, 2016
  - `std::basic_stringbuf`, 2099
- push
  - `std::priority_queue`, 2926
  - `std::queue`, 2932
  - `std::stack`, 3025
- push\_back
  - `__gnu_cxx::__versa_string`, 834
  - `__gnu_debug::basic_string`, 1079
  - `std::basic_string`, 2073
  - `std::deque`, 2395
  - `std::list`, 2624
  - `std::vector`, 3169
- push\_front
  - `__gnu_parallel::__-`  
     RestrictedBoundedConcurrentQueue,  
     1202
  - `std::deque`, 2395
  - `std::forward_list`, 2451
  - `std::list`, 2624
- push\_heap
  - heap\_algorithms, 265
- put
  - `std::basic_fstream`, 1585
  - `std::basic_istream`, 1734
  - `std::basic_ofstream`, 1889
  - `std::basic_ostream`, 1931
  - `std::basic_ostringstream`, 1976
  - `std::basic_stringstream`, 2158
  - `std::money_put`, 2722, 2723
  - `std::num_put`, 2831–2837
  - `std::time_put`, 3067, 3068
  - `std::time_put_byname`, 3071, 3072
- put\_money
  - std, 674
- putback
  - `std::basic_fstream`, 1586
  - `std::basic_ifstream`, 1641
  - `std::basic_istream`, 1734
  - `std::basic_istream`, 1787
  - `std::basic_istreamstream`, 1842
  - `std::basic_stringstream`, 2158
- pword
  - `std::basic_fstream`, 1586
  - `std::basic_ifstream`, 1642
  - `std::basic_ios`, 1677
  - `std::basic_istream`, 1735
  - `std::basic_istream`, 1788
  - `std::basic_istreamstream`, 1843
  - `std::basic_ofstream`, 1890
  - `std::basic_ostream`, 1932
  - `std::basic_ostringstream`, 1976
  - `std::basic_stringstream`, 2159
  - `std::ios_base`, 2560
- qsb\_steals
  - `__gnu_parallel::__Settings`, 1211
- queue, 3452
  - `std::queue`, 2931
- queue.h, 3453

- [\\_GLIBCXX\\_VOLATILE](#), [3453](#)
- [quicksort.h](#), [3454](#)
- [quiet\\_NaN](#)
  - [std::numeric\\_limits](#), [2841](#)
- [radix](#)
  - [std::\\_\\_numeric\\_limits\\_base](#), [1398](#)
  - [std::numeric\\_limits](#), [2845](#)
- [random](#), [3454](#)
  - [generate\\_canonical](#), [214](#)
- [Random Number Distributions](#), [283](#)
- [Random Number Generation](#), [213](#)
- [Random Number Generators](#), [276](#)
- [Random Number Utilities](#), [304](#)
- [random.h](#), [3455](#)
- [random.tcc](#), [3462](#)
- [random\\_distributions\\_bernoulli](#)
  - [operator<<](#), [294](#)
  - [operator>>](#), [295](#), [296](#)
  - [operator==](#), [295](#)
- [random\\_distributions\\_normal](#)
  - [operator<<](#), [290](#)
  - [operator>>](#), [291](#)
  - [operator==](#), [291](#)
- [random\\_distributions\\_poisson](#)
  - [operator<<](#), [300](#), [301](#)
  - [operator>>](#), [303](#), [304](#)
  - [operator==](#), [301](#), [302](#)
- [random\\_distributions\\_uniform](#)
  - [operator<<](#), [285](#), [286](#)
  - [operator>>](#), [287](#)
  - [operator==](#), [286](#)
- [random\\_generators](#)
  - [minstd\\_rand](#), [279](#)
  - [minstd\\_rand0](#), [279](#)
  - [mt19937](#), [279](#)
  - [mt19937\\_64](#), [279](#)
  - [operator<<](#), [282](#)
- [random\\_number.h](#), [3467](#)
- [random\\_sample](#)
  - [SGlextensions](#), [23](#), [24](#)
- [random\\_sample\\_n](#)
  - [SGlextensions](#), [24](#)
- [random\\_shuffle](#)
  - [mutating\\_algorithms](#), [136](#), [137](#)
- [random\\_shuffle.h](#), [3467](#)
- [random\\_shuffle\\_minimal\\_n](#)
  - [\\_\\_gnu\\_parallel::\\_Settings](#), [1211](#)
- [ratio](#), [3469](#)
- [Rational Arithmetic](#), [73](#)
- [rb\\_tree](#), [3470](#)
- [rbegin](#)
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#), [835](#)
  - [\\_\\_gnu\\_debug::basic\\_string](#), [1079](#)
  - [std::basic\\_string](#), [2073](#), [2074](#)
  - [std::deque](#), [2396](#)
  - [std::list](#), [2624](#)
  - [std::map](#), [2674](#)
  - [std::multimap](#), [2765](#)
  - [std::multiset](#), [2784](#)
  - [std::set](#), [2994](#)
  - [std::vector](#), [3169](#)
- [rc\\_string\\_base.h](#), [3471](#)
- [rdbuf](#)
  - [std::basic\\_fstream](#), [1587](#)
  - [std::basic\\_ifstream](#), [1642](#)
  - [std::basic\\_ios](#), [1677](#), [1678](#)
  - [std::basic\\_iostream](#), [1735](#), [1736](#)
  - [std::basic\\_istream](#), [1788](#), [1789](#)
  - [std::basic\\_istreamstream](#), [1843](#)
  - [std::basic\\_ofstream](#), [1890](#)
  - [std::basic\\_ostream](#), [1932](#), [1933](#)
  - [std::basic\\_ostringstream](#), [1977](#), [1978](#)
  - [std::basic\\_stringstream](#), [2159](#), [2160](#)
- [rdstate](#)
  - [std::basic\\_fstream](#), [1588](#)
  - [std::basic\\_ifstream](#), [1643](#)
  - [std::basic\\_ios](#), [1678](#)
  - [std::basic\\_iostream](#), [1736](#)
  - [std::basic\\_istream](#), [1789](#)
  - [std::basic\\_istreamstream](#), [1844](#)
  - [std::basic\\_ofstream](#), [1891](#)
  - [std::basic\\_ostream](#), [1933](#)
  - [std::basic\\_ostringstream](#), [1978](#)
  - [std::basic\\_stringstream](#), [2160](#)
- [read](#)
  - [std::basic\\_fstream](#), [1588](#)
  - [std::basic\\_ifstream](#), [1643](#)
  - [std::basic\\_iostream](#), [1737](#)
  - [std::basic\\_istream](#), [1790](#)
  - [std::basic\\_istreamstream](#), [1844](#)
  - [std::basic\\_stringstream](#), [2161](#)

- readsome
  - std::basic\_fstream, 1589
  - std::basic\_ifstream, 1644
  - std::basic\_iostream, 1737
  - std::basic\_istream, 1790
  - std::basic\_istream, 1845
  - std::basic\_stringstream, 2161
- ref
  - std, 675
- reference
  - std::back\_insert\_iterator, 1501
  - std::front\_insert\_iterator, 2460
  - std::insert\_iterator, 2548
  - std::istream\_iterator, 2583
  - std::istreambuf\_iterator, 2587
  - std::iterator, 2590
  - std::ostream\_iterator, 2887
  - std::ostreambuf\_iterator, 2891
  - std::raw\_storage\_iterator, 2944
  - std::reverse\_iterator, 2969
  - std::set, 2981
- regex, 3471
  - cregex\_token\_iterator, 220
  - csub\_match, 220
  - isctype, 222
  - operator<, 225–228
  - operator<=, 228
  - operator<=, 228–231
  - operator>, 234–237
  - operator>=, 237–239
  - operator==, 231–234
  - regex, 220
  - regex\_match, 240–243
  - regex\_replace, 244
  - regex\_search, 245–249
  - sregex\_token\_iterator, 221
  - ssub\_match, 221
  - swap, 249, 250
  - value, 250
  - wcregex\_token\_iterator, 221
  - wcsub\_match, 221
  - wregex, 221
  - wsregex\_token\_iterator, 221
  - wssub\_match, 221
- regex.h, 3471
- regex\_compiler.h, 3478
- regex\_constants.h, 3478
- regex\_cursor.h, 3480
- regex\_error
  - std::regex\_error, 2949
- regex\_error.h, 3480
- regex\_grep\_matcher.h, 3481
- regex\_grep\_matcher.tcc, 3482
- regex\_iterator
  - std::regex\_iterator, 2951
- regex\_match
  - regex, 240–243
- regex\_nfa.h, 3482
- regex\_nfa.tcc, 3483
- regex\_replace
  - regex, 244
- regex\_search
  - regex, 245–249
- regex\_token\_iterator
  - std::regex\_token\_iterator, 2955–2957
- regex\_traits
  - std::regex\_traits, 2961
- register\_callback
  - std::basic\_fstream, 1589
  - std::basic\_ifstream, 1645
  - std::basic\_ios, 1678
  - std::basic\_iostream, 1738
  - std::basic\_istream, 1791
  - std::basic\_istream, 1846
  - std::basic\_ofstream, 1891
  - std::basic\_ostream, 1934
  - std::basic\_ostream, 1978
  - std::basic\_stringstream, 2162
  - std::ios\_base, 2560
- Regular Expressions, 214
- release
  - std::auto\_ptr, 1497
- remove
  - mutating\_algorithms, 137
  - std::forward\_list, 2451
  - std::list, 2625
- remove\_copy
  - mutating\_algorithms, 138
- remove\_copy\_if
  - mutating\_algorithms, 138
- remove\_if

- mutating\_algorithms, 139
- std::forward\_list, 2452
- std::list, 2625
- rend
  - \_\_gnu\_cxx::\_\_versa\_string, 835
  - \_\_gnu\_debug::basic\_string, 1079
  - std::basic\_string, 2074
  - std::deque, 2396
  - std::list, 2625, 2626
  - std::map, 2674, 2675
  - std::multimap, 2765, 2766
  - std::multiset, 2784
  - std::set, 2995
  - std::vector, 3169
- replace
  - \_\_gnu\_cxx::\_\_versa\_string, 836–842
  - \_\_gnu\_debug::basic\_string, 1080–1085
  - mutating\_algorithms, 140
  - std::basic\_string, 2074–2080
- replace\_copy
  - std, 675
- replace\_copy\_if
  - mutating\_algorithms, 140
- replace\_if
  - mutating\_algorithms, 141
- replace\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1211
- requested\_size
  - \_\_gnu\_cxx::temporary\_buffer, 1002
  - std::\_Temporary\_buffer, 1464
- reserve
  - \_\_gnu\_cxx::\_\_versa\_string, 843
  - \_\_gnu\_debug::basic\_string, 1086
  - std::basic\_string, 2081
  - std::vector, 3170
- reset
  - std::auto\_ptr, 1497
  - std::bernoulli\_distribution, 2181
  - std::binomial\_distribution, 2192
  - std::bitset, 2206
  - std::cauchy\_distribution, 2211
  - std::chi\_squared\_distribution, 2220
  - std::discrete\_distribution, 2408
  - std::exponential\_distribution, 2423
  - std::extreme\_value\_distribution, 2427
  - std::fisher\_f\_distribution, 2431
  - std::gamma\_distribution, 2483
  - std::geometric\_distribution, 2488
  - std::lognormal\_distribution, 2654
  - std::negative\_binomial\_distribution, 2792
  - std::normal\_distribution, 2799
  - std::piecewise\_constant\_distribution, 2904
  - std::piecewise\_linear\_distribution, 2910
  - std::poisson\_distribution, 2920
  - std::student\_t\_distribution, 3029
  - std::uniform\_int\_distribution, 3125
  - std::uniform\_real\_distribution, 3129
  - std::weibull\_distribution, 3179
- resetiosflags
  - std, 676
- resize
  - \_\_gnu\_cxx::\_\_versa\_string, 844
  - \_\_gnu\_debug::basic\_string, 1086, 1087
  - numeric\_arrays, 106
  - std::basic\_string, 2082
  - std::deque, 2397
  - std::forward\_list, 2452
  - std::list, 2626
  - std::vector, 3170, 3171
- result\_type
  - \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder, 772
  - \_\_gnu\_cxx::binary\_compose, 863
  - \_\_gnu\_cxx::project1st, 920
  - \_\_gnu\_cxx::project2nd, 921
  - \_\_gnu\_cxx::select1st, 936
  - \_\_gnu\_cxx::select2nd, 937
  - \_\_gnu\_cxx::subtractive\_rng, 999
  - \_\_gnu\_cxx::unary\_compose, 1013
  - \_\_gnu\_parallel::\_EqualFromLess, 1143
  - \_\_gnu\_parallel::\_EqualTo, 1145
  - \_\_gnu\_parallel::\_Less, 1155
  - \_\_gnu\_parallel::\_Lexicographic, 1157

- `__gnu_parallel::_-`  
LexicographicReverse, 1159
- `__gnu_parallel::_Multiplies`, 1187
- `__gnu_parallel::_Plus`, 1191
- `__gnu_parallel::__binder1st`, 1098
- `__gnu_parallel::__binder2nd`, 1100
- `__gnu_parallel::__unary_negate`, 1136
- `std::_Maybe_unary_or_binary_-`  
function< \_Res, \_T1 >, 1449
- `std::_Maybe_unary_or_binary_-`  
function< \_Res, \_T1, \_T2 >, 1451
- `std::bernoulli_distribution`, 2179
- `std::binary_function`, 2184
- `std::binary_negate`, 2186
- `std::binder1st`, 2187
- `std::binder2nd`, 2189
- `std::binomial_distribution`, 2191
- `std::cauchy_distribution`, 2210
- `std::chi_squared_distribution`, 2218
- `std::const_mem_fun1_ref_t`, 2275
- `std::const_mem_fun1_t`, 2277
- `std::const_mem_fun_ref_t`, 2279
- `std::const_mem_fun_t`, 2281
- `std::discard_block_engine`, 2400
- `std::discrete_distribution`, 2406
- `std::divides`, 2411
- `std::equal_to`, 2416
- `std::exponential_distribution`, 2421
- `std::extreme_value_distribution`, 2425
- `std::fisher_f_distribution`, 2429
- `std::gamma_distribution`, 2481
- `std::geometric_distribution`, 2486
- `std::greater`, 2490
- `std::greater_equal`, 2492
- `std::hash`, 2500
- `std::hash< __debug::bitset< _Nb >`  
>, 2502
- `std::hash< __debug::vector< bool,`  
\_Alloc > >, 2504
- `std::hash< __gnu_cxx::throw_-`  
value\_limit >, 2506
- `std::hash< __gnu_cxx::throw_-`  
value\_random >, 2508
- `std::hash< __profile::bitset< _Nb >`  
>, 2510
- `std::hash< __profile::vector< bool,`  
\_Alloc > >, 2512
- `std::hash< _Tp * >`, 2514
- `std::hash< error_code >`, 2516
- `std::hash< shared_ptr< _Tp > >`, 2518
- `std::hash< string >`, 2520
- `std::hash< thread::id >`, 2522
- `std::hash< u16string >`, 2524
- `std::hash< u32string >`, 2526
- `std::hash< unique_ptr< _Tp, _Tp_-`  
Deleter > >, 2528
- `std::hash< wstring >`, 2530
- `std::hash<::bitset< _Nb > >`, 2532
- `std::hash<::vector< bool, _Alloc >`  
>, 2534
- `std::independent_bits_engine`, 2536
- `std::less`, 2597
- `std::less_equal`, 2599
- `std::linear_congruential_engine`, 2601
- `std::logical_and`, 2647
- `std::logical_not`, 2649
- `std::logical_or`, 2651
- `std::lognormal_distribution`, 2653
- `std::mem_fun1_ref_t`, 2692
- `std::mem_fun1_t`, 2694
- `std::mem_fun_ref_t`, 2696
- `std::mem_fun_t`, 2698
- `std::minus`, 2708
- `std::modulus`, 2710
- `std::multiplies`, 2769
- `std::negate`, 2789
- `std::negative_binomial_distribution`, 2791
- `std::normal_distribution`, 2797
- `std::not_equal_to`, 2803
- `std::piecewise_constant_-`  
distribution, 2903
- `std::piecewise_linear_distribution`, 2908
- `std::plus`, 2913
- `std::pointer_to_binary_function`, 2915



- 
- std::pointer\_to\_unary\_function, 2917
  - std::poisson\_distribution, 2918
  - std::random\_device, 2935
  - std::seed\_seq, 2976
  - std::shuffle\_order\_engine, 3014
  - std::student\_t\_distribution, 3028
  - std::unary\_function, 3119
  - std::unary\_negate, 3121
  - std::uniform\_int\_distribution, 3124
  - std::uniform\_real\_distribution, 3128
  - std::weibull\_distribution, 3178
  - rethrow\_exception
    - exceptions, 36
  - rethrow\_if\_nested
    - exceptions, 37
  - return\_temporary\_buffer
    - std, 676
  - reverse
    - mutating\_algorithms, 141
    - std::forward\_list, 2453
    - std::list, 2626
  - reverse\_copy
    - mutating\_algorithms, 142
  - reverse\_iterator
    - std::reverse\_iterator, 2969, 2970
    - std::set, 2981
  - rfind
    - \_\_gnu\_cxx::\_\_versa\_string, 844–846
    - \_\_gnu\_debug::basic\_string, 1087, 1088
    - std::basic\_string, 2082–2084
  - riemann\_zeta
    - tr1\_math\_spec\_func, 114
  - right
    - std, 676
    - std::basic\_fstream, 1604
    - std::basic\_ifstream, 1657
    - std::basic\_ios, 1688
    - std::basic\_iostream, 1753
    - std::basic\_istream, 1804
    - std::basic\_istream, 1859
    - std::basic\_ofstream, 1903
    - std::basic\_ostream, 1945
    - std::basic\_ostringstream, 1990
    - std::basic\_stringstream, 2177
    - std::ios\_base, 2568
  - rope, 3483
  - ropeimpl.h, 3488
  - rotate
    - mutating\_algorithms, 142
  - rotate\_copy
    - mutating\_algorithms, 143
  - round\_indeterminate
    - std, 600
  - round\_to\_nearest
    - std, 600
  - round\_toward\_infinity
    - std, 600
  - round\_toward\_neg\_infinity
    - std, 600
  - round\_toward\_zero
    - std, 600
  - round\_error
    - std::numeric\_limits, 2841
  - round\_style
    - std::\_\_numeric\_limits\_base, 1398
    - std::numeric\_limits, 2845
  - runtime\_error
    - std::runtime\_error, 2975
  - safe\_base.h, 3488
  - safe\_iterator.h, 3489
  - safe\_iterator.tcc, 3491
  - safe\_sequence.h, 3491
  - sbumpc
    - \_\_gnu\_cxx::enc\_filebuf, 887
    - \_\_gnu\_cxx::stdio\_filebuf, 958
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 986
    - std::basic\_filebuf, 1524
    - std::basic\_streambuf, 2017
    - std::basic\_stringbuf, 2099
  - scan\_is
    - std::\_\_ctype\_abstract\_base, 1320
    - std::ctype, 2293
    - std::ctype< char >, 2306
    - std::ctype< wchar\_t >, 2328
    - std::ctype\_byname, 2344
    - std::ctype\_byname< char >, 2356
  - scan\_not
    - std::\_\_ctype\_abstract\_base, 1320
-

- std::ctype, 2293
- std::ctype< char >, 2307
- std::ctype< wchar\_t >, 2328
- std::ctype\_byname, 2344
- std::ctype\_byname< char >, 2357
- scientific
  - std, 677
  - std::basic\_fstream, 1604
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1688
  - std::basic\_iostream, 1753
  - std::basic\_istream, 1804
  - std::basic\_istream, 1859
  - std::basic\_ofstream, 1903
  - std::basic\_ostream, 1945
  - std::basic\_ostringstream, 1990
  - std::basic\_stringstream, 2177
  - std::ios\_base, 2568
- search
  - non\_mutating\_algorithms, 160, 161
- search.h, 3491
- search\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1211
- search\_n
  - non\_mutating\_algorithms, 162
- second
  - \_\_gnu\_parallel::\_IteratorPair, 1150
  - std::pair, 2901
  - std::sub\_match, 3036
- second\_argument\_type
  - \_\_gnu\_cxx::project1st, 920
  - \_\_gnu\_cxx::project2nd, 921
  - \_\_gnu\_parallel::\_EqualFromLess, 1143
  - \_\_gnu\_parallel::\_EqualTo, 1145
  - \_\_gnu\_parallel::\_Less, 1155
  - \_\_gnu\_parallel::\_Lexicographic, 1157
  - \_\_gnu\_parallel::\_-
    - LexicographicReverse, 1159
  - \_\_gnu\_parallel::\_Multiplies, 1187
  - \_\_gnu\_parallel::\_Plus, 1191
  - std::\_Maybe\_unary\_or\_binary\_-
    - function< \_Res, \_T1, \_T2 >, 1451
  - std::binary\_function, 2184
  - std::binary\_negate, 2186
  - std::const\_mem\_fun1\_ref\_t, 2275
  - std::const\_mem\_fun1\_t, 2277
  - std::divides, 2411
  - std::equal\_to, 2416
  - std::greater, 2490
  - std::greater\_equal, 2492
  - std::less, 2597
  - std::less\_equal, 2599
  - std::logical\_and, 2647
  - std::logical\_or, 2651
  - std::mem\_fun1\_ref\_t, 2692
  - std::mem\_fun1\_t, 2694
  - std::minus, 2708
  - std::modulus, 2710
  - std::multiplies, 2769
  - std::not\_equal\_to, 2803
  - std::plus, 2913
  - std::pointer\_to\_binary\_function, 2915
- second\_type
  - \_\_gnu\_parallel::\_IteratorPair, 1150
  - std::pair, 2900
  - std::sub\_match, 3033
- seconds
  - std::chrono, 729
- seed
  - std::discard\_block\_engine, 2403
  - std::independent\_bits\_engine, 2539
  - std::linear\_congruential\_engine, 2603
  - std::shuffle\_order\_engine, 3017
- seed\_seq
  - std::seed\_seq, 2976
- seekdir
  - std::basic\_fstream, 1552
  - std::basic\_ifstream, 1616
  - std::basic\_ios, 1667
  - std::basic\_iostream, 1702
  - std::basic\_istream, 1764
  - std::basic\_istream, 1818
  - std::basic\_ofstream, 1870
  - std::basic\_ostream, 1914
  - std::basic\_ostringstream, 1958
  - std::basic\_stringstream, 2126
  - std::ios\_base, 2556

- seekg
  - std::basic\_fstream, 1590
  - std::basic\_ifstream, 1645, 1646
  - std::basic\_iostream, 1738, 1739
  - std::basic\_istream, 1791, 1792
  - std::basic\_istreamstream, 1846, 1847
  - std::basic\_stringstream, 2162, 2163
- seekoff
  - \_\_gnu\_cxx::enc\_filebuf, 887
  - \_\_gnu\_cxx::stdio\_filebuf, 959
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 987
  - std::basic\_filebuf, 1524
  - std::basic\_streambuf, 2017
  - std::basic\_stringbuf, 2100
- seekp
  - std::basic\_fstream, 1591
  - std::basic\_iostream, 1740
  - std::basic\_ofstream, 1892
  - std::basic\_ostream, 1934
  - std::basic\_ostreamstream, 1978, 1979
  - std::basic\_stringstream, 2164
- seekpos
  - \_\_gnu\_cxx::enc\_filebuf, 887
  - \_\_gnu\_cxx::stdio\_filebuf, 959
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 987
  - std::basic\_filebuf, 1525
  - std::basic\_streambuf, 2017
  - std::basic\_stringbuf, 2100
- sentry
  - std::basic\_istream::sentry, 1806
  - std::basic\_ostream::sentry, 1947
- Sequences, 27
- sequential
  - \_\_gnu\_parallel, 403
- set, 3492
  - \_\_gnu\_parallel::\_Settings, 1206
  - std::bitset, 2206
  - std::set, 2982–2984
- Set Operation, 186
- set.h, 3493, 3494
- set\_algorithms
  - includes, 187
  - set\_difference, 188, 189
  - set\_intersection, 189, 190
  - set\_symmetric\_difference, 191
  - set\_union, 192
- set\_difference
  - set\_algorithms, 188, 189
- set\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1211
- set\_intersection
  - set\_algorithms, 189, 190
- set\_intersection\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1211
- set\_new\_handler
  - std, 677
- set\_num\_threads
  - \_\_gnu\_parallel::balanced\_-quicksort\_tag, 1216
  - \_\_gnu\_parallel::balanced\_tag, 1218
  - \_\_gnu\_parallel::default\_parallel\_tag, 1220
  - \_\_gnu\_parallel::exact\_tag, 1223
  - \_\_gnu\_parallel::multiway\_-mergesort\_exact\_tag, 1226
  - \_\_gnu\_parallel::multiway\_-mergesort\_sampling\_tag, 1228
  - \_\_gnu\_parallel::multiway\_-mergesort\_tag, 1229
  - \_\_gnu\_parallel::omp\_loop\_static\_tag, 1231
  - \_\_gnu\_parallel::omp\_loop\_tag, 1232
  - \_\_gnu\_parallel::parallel\_tag, 1235
  - \_\_gnu\_parallel::quicksort\_tag, 1237
  - \_\_gnu\_parallel::sampling\_tag, 1238
  - \_\_gnu\_parallel::unbalanced\_tag, 1240
- set\_operations.h, 3495
- set\_symmetric\_difference
  - set\_algorithms, 191
- set\_symmetric\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1212
- set\_terminate
  - exceptions, 37
- set\_unexpected
  - exceptions, 37
- set\_union
  - set\_algorithms, 192
- set\_union\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1212

- setbase
  - std, 677
- setbuf
  - \_\_gnu\_cxx::enc\_filebuf, 888
  - \_\_gnu\_cxx::stdio\_filebuf, 960
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 988
  - std::basic\_filebuf, 1525, 1526
  - std::basic\_streambuf, 2018
  - std::basic\_stringbuf, 2101
- setf
  - std::basic\_fstream, 1592
  - std::basic\_ifstream, 1646, 1647
  - std::basic\_ios, 1679
  - std::basic\_iostream, 1741
  - std::basic\_istream, 1793
  - std::basic\_istream, 1847, 1848
  - std::basic\_ofstream, 1893
  - std::basic\_ostream, 1935
  - std::basic\_ostringstream, 1979, 1980
  - std::basic\_stringstream, 2165
  - std::ios\_base, 2561
- setfill
  - std, 677
- setg
  - \_\_gnu\_cxx::enc\_filebuf, 888
  - \_\_gnu\_cxx::stdio\_filebuf, 960
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 988
  - std::basic\_filebuf, 1526
  - std::basic\_streambuf, 2018
  - std::basic\_stringbuf, 2101
- setiosflags
  - std, 677
- setp
  - \_\_gnu\_cxx::enc\_filebuf, 889
  - \_\_gnu\_cxx::stdio\_filebuf, 961
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 988
  - std::basic\_filebuf, 1526
  - std::basic\_streambuf, 2019
  - std::basic\_stringbuf, 2102
- setprecision
  - std, 678
- setstate
  - std::basic\_fstream, 1593
  - std::basic\_ifstream, 1647
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1741
  - std::basic\_istream, 1794
  - std::basic\_istream, 1848
  - std::basic\_ofstream, 1893
  - std::basic\_ostream, 1936
  - std::basic\_ostringstream, 1980
  - std::basic\_stringstream, 2165
- settings.h, 3496
  - \_GLIBCXX\_PARALLEL\_CONDITION, 3497
- setw
  - std, 678
- sgetc
  - \_\_gnu\_cxx::enc\_filebuf, 889
  - \_\_gnu\_cxx::stdio\_filebuf, 961
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 989
  - std::basic\_filebuf, 1527
  - std::basic\_streambuf, 2019
  - std::basic\_stringbuf, 2102
- sgetn
  - \_\_gnu\_cxx::enc\_filebuf, 889
  - \_\_gnu\_cxx::stdio\_filebuf, 962
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 989
  - std::basic\_filebuf, 1527
  - std::basic\_streambuf, 2019
  - std::basic\_stringbuf, 2103
- SGL, 12
- SGLextensions
  - \_Find\_first, 17
  - \_Find\_next, 17
  - \_Unchecked\_flip, 18
  - \_Unchecked\_reset, 18
  - \_Unchecked\_set, 18
  - \_Unchecked\_test, 18
  - \_\_median, 16, 17
  - compose1, 19
  - compose2, 19
  - constant0, 19
  - constant1, 19
  - constant2, 19
  - copy\_n, 20
  - distance, 20
  - identity\_element, 20, 21
  - iota, 21
  - is\_heap, 21
  - is\_sorted, 22
  - lexicographical\_compare\_3way, 22

- power, 23
- random\_sample, 23, 24
- random\_sample\_n, 24
- uninitialized\_copy\_n, 25
- shared\_future
  - std::shared\_future, 2998
  - std::shared\_future< \_Res & >, 3001
  - std::shared\_future< void >, 3004
- shared\_ptr
  - std::shared\_ptr, 3007–3011
- shared\_ptr.h, 3497
- shared\_ptr\_base.h, 3499
- shift
  - numeric\_arrays, 106
- showbase
  - std, 678
  - std::basic\_fstream, 1604
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1688
  - std::basic\_iostream, 1753
  - std::basic\_istream, 1804
  - std::basic\_istreamstream, 1859
  - std::basic\_ofstream, 1903
  - std::basic\_ostream, 1945
  - std::basic\_ostreamstream, 1990
  - std::basic\_stringstream, 2177
  - std::ios\_base, 2568
- showmanyc
  - \_\_gnu\_cxx::enc\_filebuf, 890
  - \_\_gnu\_cxx::stdio\_filebuf, 962
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 989
  - std::basic\_filebuf, 1528
  - std::basic\_streambuf, 2020
  - std::basic\_stringbuf, 2103
- showpoint
  - std, 678
  - std::basic\_fstream, 1604
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1688
  - std::basic\_iostream, 1753
  - std::basic\_istream, 1804
  - std::basic\_istreamstream, 1859
  - std::basic\_ofstream, 1903
  - std::basic\_ostream, 1945
  - std::basic\_ostreamstream, 1991
  - std::basic\_stringstream, 2177
  - std::ios\_base, 2568
- showpos
  - std, 679
  - std::basic\_fstream, 1604
  - std::basic\_ifstream, 1658
  - std::basic\_ios, 1688
  - std::basic\_iostream, 1753
  - std::basic\_istream, 1804
  - std::basic\_istreamstream, 1859
  - std::basic\_ofstream, 1903
  - std::basic\_ostream, 1946
  - std::basic\_ostreamstream, 1991
  - std::basic\_stringstream, 2178
  - std::ios\_base, 2568
- shrink\_to\_fit
  - \_\_gnu\_cxx::\_\_versa\_string, 847
  - \_\_gnu\_debug::basic\_string, 1089
  - std::basic\_string, 2084
  - std::deque, 2397
  - std::vector, 3171
- shuffle
  - mutating\_algorithms, 143
- shuffle\_order\_engine
  - std::shuffle\_order\_engine, 3014, 3015
- signaling\_NaN
  - std::numeric\_limits, 2841
- sin
  - complex\_numbers, 53
- sinh
  - complex\_numbers, 53
- size
  - \_\_gnu\_cxx::\_\_versa\_string, 847
  - \_\_gnu\_cxx::temporary\_buffer, 1002
  - \_\_gnu\_debug::basic\_string, 1089
  - numeric\_arrays, 106, 107
  - std::\_Temporary\_buffer, 1465
  - std::basic\_string, 2085
  - std::bitset, 2207
  - std::deque, 2397
  - std::list, 2627
  - std::map, 2675
  - std::match\_results, 2689
  - std::multimap, 2766
  - std::multiset, 2784
  - std::priority\_queue, 2926

- std::queue, 2932
- std::set, 2995
- std::stack, 3026
- std::vector, 3171
- size\_type
  - std::set, 2982
- skipws
  - std, 679
  - std::basic\_fstream, 1605
  - std::basic\_ifstream, 1658
  - std::basic\_ios, 1689
  - std::basic\_iostream, 1753
  - std::basic\_istream, 1804
  - std::basic\_istream, 1859
  - std::basic\_ofstream, 1904
  - std::basic\_ostream, 1946
  - std::basic\_ostringstream, 1991
  - std::basic\_stringstream, 2178
  - std::ios\_base, 2569
- sleep\_for
  - std::this\_thread, 753
- sleep\_until
  - std::this\_thread, 754
- slice
  - numeric\_arrays, 88
- slice\_array
  - numeric\_arrays, 88
- slice\_array.h, 3499
- slist, 3500
- snextc
  - \_\_gnu\_cxx::enc\_filebuf, 890
  - \_\_gnu\_cxx::stdio\_filebuf, 963
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 990
  - std::basic\_filebuf, 1528
  - std::basic\_streambuf, 2020
  - std::basic\_stringbuf, 2104
- sort
  - sorting\_algorithms, 183, 184
  - std::forward\_list, 2453
  - std::list, 2627
- sort.h, 3501
- sort\_heap
  - heap\_algorithms, 266
- sort\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1212
- sort\_mwms\_oversampling
  - \_\_gnu\_parallel::\_Settings, 1212
- sort\_qs\_num\_samples\_preset
  - \_\_gnu\_parallel::\_Settings, 1212
- sort\_qsb\_base\_case\_maximal\_n
  - \_\_gnu\_parallel::\_Settings, 1212
- Sorting, 164
- sorting\_algorithms
  - inplace\_merge, 167
  - is\_sorted, 168
  - is\_sorted\_until, 169
  - lexicographical\_compare, 170
  - max, 171
  - max\_element, 172
  - merge, 173
  - min, 174
  - min\_element, 175
  - minmax, 176
  - minmax\_element, 176, 177
  - next\_permutation, 177, 178
  - nth\_element, 178, 179
  - partial\_sort, 180
  - partial\_sort\_copy, 181
  - prev\_permutation, 182, 183
  - sort, 183, 184
  - stable\_sort, 184, 185
- sph\_bessel
  - tr1\_math\_spec\_func, 114
- sph\_legendre
  - tr1\_math\_spec\_func, 114
- sph\_neumann
  - tr1\_math\_spec\_func, 115
- splice
  - std::list, 2627, 2628
- splice\_after
  - std::forward\_list, 2453, 2454
- sputbackc
  - \_\_gnu\_cxx::enc\_filebuf, 891
  - \_\_gnu\_cxx::stdio\_filebuf, 963
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 990
  - std::basic\_filebuf, 1529
  - std::basic\_streambuf, 2021
  - std::basic\_stringbuf, 2104
- sputc
  - \_\_gnu\_cxx::enc\_filebuf, 891
  - \_\_gnu\_cxx::stdio\_filebuf, 963
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 991

- 
- std::basic\_filebuf, 1529
  - std::basic\_streambuf, 2021
  - std::basic\_stringbuf, 2104
  - sputn
    - \_\_gnu\_cxx::enc\_filebuf, 891
    - \_\_gnu\_cxx::stdio\_filebuf, 964
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 991
  - std::basic\_filebuf, 1530
  - std::basic\_streambuf, 2022
  - std::basic\_stringbuf, 2105
  - sqrt
    - complex\_numbers, 53
  - sregex\_token\_iterator
    - regex, 221
  - sso\_string\_base.h, 3502
  - sstream, 3503
  - sstream.tcc, 3504
  - ssub\_match
    - regex, 221
  - stable\_partition
    - mutating\_algorithms, 144
  - stable\_sort
    - sorting\_algorithms, 184, 185
  - stack, 3504
    - std::stack, 3025
  - standard\_policies.hpp, 3504
  - start
    - numeric\_arrays, 107
  - state
    - std::fpos, 2458
  - std, 460
    - \_Construct, 614
    - \_Destroy, 614
    - \_final\_insertion\_sort, 601
    - \_find, 601
    - \_find\_if, 601, 602
    - \_find\_if\_not, 602
    - \_gcd, 602
    - \_heap\_select, 602, 603
    - \_inplace\_stable\_partition, 603
    - \_inplace\_stable\_sort, 603
    - \_insertion\_sort, 604
    - \_introsort\_loop, 604
    - \_invoke, 685
    - \_lg, 605
    - \_merge\_adaptive, 605
    - \_merge\_backward, 606
    - \_merge\_without\_buffer, 606
    - \_move\_median\_first, 607
    - \_partition, 607
    - \_reverse, 608
    - \_rotate, 608, 609
    - \_rotate\_adaptive, 609
    - \_search\_n, 609, 610
    - \_stable\_partition\_adaptive, 610
    - \_unguarded\_insertion\_sort, 610, 611
    - \_unguarded\_linear\_insert, 611
    - \_unguarded\_partition, 611, 612
    - \_unguarded\_partition\_pivot, 612
    - \_unique\_copy, 612–614
  - accumulate, 615
  - addressof, 615
  - adjacent\_difference, 616
  - advance, 617
  - bind, 617
  - boolalpha, 618
  - cerr, 685
  - cin, 686
  - clog, 686
  - cout, 686
  - cref, 618
  - dec, 618
  - denorm\_absent, 600
  - denorm\_indeterminate, 600
  - denorm\_present, 600
  - distance, 618
  - endl, 619
  - ends, 619
  - fixed, 620
  - float\_denorm\_style, 600
  - float\_round\_style, 600
  - flush, 620
  - forward, 620
  - get\_money, 621
  - get\_temporary\_buffer, 621
  - getline, 621–623
  - has\_facet, 624
  - hex, 624
  - inner\_product, 624, 625
  - internal, 626
  - iota, 626
-

- isalnum, [626](#)
- isalpha, [626](#)
- isctrl, [626](#)
- isdigit, [627](#)
- isgraph, [627](#)
- islower, [627](#)
- isprint, [627](#)
- ispunct, [627](#)
- isspace, [627](#)
- isupper, [627](#)
- isxdigit, [627](#)
- left, [628](#)
- make\_pair, [628](#)
- new\_handler, [599](#)
- noboolalpha, [628](#)
- noshowbase, [629](#)
- noshowpoint, [629](#)
- noshowpos, [629](#)
- noskipws, [629](#)
- nunitbuf, [629](#)
- nuppercase, [629](#)
- oct, [630](#)
- operator<, [636–642](#)
- operator<=, [642–648](#)
- operator<=, [649–652](#)
- operator>, [659–662](#)
- operator>=, [666–672](#)
- operator>=, [662–666](#)
- operator^, [673](#)
- operator+, [634–636](#)
- operator==, [652–659](#)
- operator&, [634](#)
- partial\_sum, [673, 674](#)
- put\_money, [674](#)
- ref, [675](#)
- replace\_copy, [675](#)
- resetiosflags, [676](#)
- return\_temporary\_buffer, [676](#)
- right, [676](#)
- round\_indeterminate, [600](#)
- round\_to\_nearest, [600](#)
- round\_toward\_infinity, [600](#)
- round\_toward\_neg\_infinity, [600](#)
- round\_toward\_zero, [600](#)
- scientific, [677](#)
- set\_new\_handler, [677](#)
- setbase, [677](#)
- setfill, [677](#)
- setiosflags, [677](#)
- setprecision, [678](#)
- setw, [678](#)
- showbase, [678](#)
- showpoint, [678](#)
- showpos, [679](#)
- skipws, [679](#)
- streamoff, [599](#)
- streampos, [599](#)
- streamsize, [599](#)
- swap, [679–681](#)
- tolower, [682](#)
- toupper, [682](#)
- u16streampos, [599](#)
- u32streampos, [599](#)
- uninitialized\_copy, [682](#)
- uninitialized\_copy\_n, [682](#)
- uninitialized\_fill, [683](#)
- uninitialized\_fill\_n, [683](#)
- unitbuf, [684](#)
- uppercase, [684](#)
- use\_facet, [684](#)
- wcerr, [686](#)
- wcin, [686](#)
- wclog, [686](#)
- wcout, [686](#)
- ws, [685](#)
- wstreampos, [599](#)
- std::\_\_basic\_future, [1302](#)
- std::\_\_M\_get\_result, [1303](#)
- std::\_\_codecvt\_abstract\_base, [1303](#)
- do\_out, [1305](#)
- in, [1305](#)
- out, [1306](#)
- unshift, [1307](#)
- std::\_\_ctype\_abstract\_base, [1308](#)
- char\_type, [1311](#)
- do\_is, [1312](#)
- do\_narrow, [1312, 1313](#)
- do\_scan\_is, [1314](#)
- do\_scan\_not, [1314](#)
- do\_tolower, [1315](#)
- do\_toupper, [1316](#)
- do\_widen, [1317](#)



- is, [1318](#)
- narrow, [1319](#)
- scan\_is, [1320](#)
- scan\_not, [1320](#)
- tolower, [1321](#)
- toupper, [1321](#), [1322](#)
- widen, [1322](#), [1323](#)
- std::\_\_debug, [686](#)
- std::\_\_debug::bitset, [1324](#)
  - \_M\_const\_iterators, [1327](#)
  - \_M\_detach\_all, [1326](#)
  - \_M\_detach\_singular, [1326](#)
  - \_M\_get\_mutex, [1326](#)
  - \_M\_invalidate\_all, [1326](#)
  - \_M\_iterators, [1327](#)
  - \_M\_revalidate\_singular, [1327](#)
  - \_M\_swap, [1327](#)
  - \_M\_version, [1327](#)
- std::\_\_debug::deque, [1328](#)
  - \_M\_const\_iterators, [1333](#)
  - \_M\_detach\_all, [1331](#)
  - \_M\_detach\_singular, [1331](#)
  - \_M\_get\_mutex, [1331](#)
  - \_M\_invalidate\_all, [1332](#)
  - \_M\_invalidate\_if, [1332](#)
  - \_M\_iterators, [1333](#)
  - \_M\_revalidate\_singular, [1332](#)
  - \_M\_swap, [1332](#)
  - \_M\_transfer\_iter, [1332](#)
  - \_M\_version, [1333](#)
- std::\_\_debug::list, [1333](#)
  - \_M\_const\_iterators, [1338](#)
  - \_M\_detach\_all, [1337](#)
  - \_M\_detach\_singular, [1337](#)
  - \_M\_get\_mutex, [1337](#)
  - \_M\_invalidate\_all, [1337](#)
  - \_M\_invalidate\_if, [1338](#)
  - \_M\_iterators, [1339](#)
  - \_M\_revalidate\_singular, [1338](#)
  - \_M\_swap, [1338](#)
  - \_M\_transfer\_iter, [1338](#)
  - \_M\_version, [1339](#)
- std::\_\_debug::map, [1339](#)
  - \_M\_const\_iterators, [1344](#)
  - \_M\_detach\_all, [1342](#)
  - \_M\_detach\_singular, [1342](#)
  - \_M\_get\_mutex, [1343](#)
  - \_M\_invalidate\_all, [1343](#)
  - \_M\_invalidate\_if, [1343](#)
  - \_M\_iterators, [1344](#)
  - \_M\_revalidate\_singular, [1343](#)
  - \_M\_swap, [1343](#)
  - \_M\_transfer\_iter, [1344](#)
  - \_M\_version, [1344](#)
- std::\_\_debug::multimap, [1345](#)
  - \_M\_const\_iterators, [1349](#)
  - \_M\_detach\_all, [1348](#)
  - \_M\_detach\_singular, [1348](#)
  - \_M\_get\_mutex, [1348](#)
  - \_M\_invalidate\_all, [1348](#)
  - \_M\_invalidate\_if, [1348](#)
  - \_M\_iterators, [1349](#)
  - \_M\_revalidate\_singular, [1348](#)
  - \_M\_swap, [1349](#)
  - \_M\_transfer\_iter, [1349](#)
  - \_M\_version, [1349](#)
- std::\_\_debug::multiset, [1350](#)
  - \_M\_const\_iterators, [1354](#)
  - \_M\_detach\_all, [1353](#)
  - \_M\_detach\_singular, [1353](#)
  - \_M\_get\_mutex, [1353](#)
  - \_M\_invalidate\_all, [1353](#)
  - \_M\_invalidate\_if, [1353](#)
  - \_M\_iterators, [1354](#)
  - \_M\_revalidate\_singular, [1354](#)
  - \_M\_swap, [1354](#)
  - \_M\_transfer\_iter, [1354](#)
  - \_M\_version, [1355](#)
- std::\_\_debug::set, [1355](#)
  - \_M\_const\_iterators, [1360](#)
  - \_M\_detach\_all, [1358](#)
  - \_M\_detach\_singular, [1358](#)
  - \_M\_get\_mutex, [1359](#)
  - \_M\_invalidate\_all, [1359](#)
  - \_M\_invalidate\_if, [1359](#)
  - \_M\_iterators, [1360](#)
  - \_M\_revalidate\_singular, [1359](#)
  - \_M\_swap, [1359](#)
  - \_M\_transfer\_iter, [1360](#)
  - \_M\_version, [1360](#)
- std::\_\_debug::unordered\_map, [1361](#)
  - \_M\_const\_iterators, [1365](#)

- [\\_M\\_detach\\_all, 1363](#)
- [\\_M\\_detach\\_singular, 1363](#)
- [\\_M\\_get\\_mutex, 1363](#)
- [\\_M\\_invalidate\\_all, 1364](#)
- [\\_M\\_invalidate\\_if, 1364](#)
- [\\_M\\_iterators, 1365](#)
- [\\_M\\_revalidate\\_singular, 1364](#)
- [\\_M\\_swap, 1364](#)
- [\\_M\\_transfer\\_iter, 1364](#)
- [\\_M\\_version, 1365](#)
- [std::\\_\\_debug::unordered\\_multimap, 1366](#)
  - [\\_M\\_const\\_iterators, 1369](#)
  - [\\_M\\_detach\\_all, 1368](#)
  - [\\_M\\_detach\\_singular, 1368](#)
  - [\\_M\\_get\\_mutex, 1368](#)
  - [\\_M\\_invalidate\\_all, 1368](#)
  - [\\_M\\_invalidate\\_if, 1368](#)
  - [\\_M\\_iterators, 1369](#)
  - [\\_M\\_revalidate\\_singular, 1369](#)
  - [\\_M\\_swap, 1369](#)
  - [\\_M\\_transfer\\_iter, 1369](#)
  - [\\_M\\_version, 1370](#)
- [std::\\_\\_debug::unordered\\_multiset, 1370](#)
  - [\\_M\\_const\\_iterators, 1374](#)
  - [\\_M\\_detach\\_all, 1372](#)
  - [\\_M\\_detach\\_singular, 1372](#)
  - [\\_M\\_get\\_mutex, 1373](#)
  - [\\_M\\_invalidate\\_all, 1373](#)
  - [\\_M\\_invalidate\\_if, 1373](#)
  - [\\_M\\_iterators, 1374](#)
  - [\\_M\\_revalidate\\_singular, 1373](#)
  - [\\_M\\_swap, 1373](#)
  - [\\_M\\_transfer\\_iter, 1373](#)
  - [\\_M\\_version, 1374](#)
- [std::\\_\\_debug::unordered\\_set, 1375](#)
  - [\\_M\\_const\\_iterators, 1379](#)
  - [\\_M\\_detach\\_all, 1377](#)
  - [\\_M\\_detach\\_singular, 1377](#)
  - [\\_M\\_get\\_mutex, 1377](#)
  - [\\_M\\_invalidate\\_all, 1378](#)
  - [\\_M\\_invalidate\\_if, 1378](#)
  - [\\_M\\_iterators, 1379](#)
  - [\\_M\\_revalidate\\_singular, 1378](#)
  - [\\_M\\_swap, 1378](#)
  - [\\_M\\_transfer\\_iter, 1378](#)
  - [\\_M\\_version, 1379](#)
- [std::\\_\\_debug::vector, 1380](#)
  - [\\_M\\_const\\_iterators, 1384](#)
  - [\\_M\\_detach\\_all, 1383](#)
  - [\\_M\\_detach\\_singular, 1383](#)
  - [\\_M\\_get\\_mutex, 1383](#)
  - [\\_M\\_invalidate\\_all, 1383](#)
  - [\\_M\\_invalidate\\_if, 1384](#)
  - [\\_M\\_iterators, 1385](#)
  - [\\_M\\_revalidate\\_singular, 1384](#)
  - [\\_M\\_swap, 1384](#)
  - [\\_M\\_transfer\\_iter, 1384](#)
  - [\\_M\\_version, 1385](#)
  - [vector, 1383](#)
- [std::\\_\\_declval\\_protector, 1385](#)
- [std::\\_\\_detail, 692](#)
- [std::\\_\\_exception\\_ptr::exception\\_ptr, 1386](#)
- [std::\\_\\_future\\_base, 1387](#)
- [std::\\_\\_future\\_base::\\_Ptr, 1388](#)
- [std::\\_\\_future\\_base::\\_Result, 1389](#)
- [std::\\_\\_future\\_base::\\_Result< \\_Res & >, 1390](#)
- [std::\\_\\_future\\_base::\\_Result< void >, 1391](#)
- [std::\\_\\_future\\_base::\\_Result\\_base, 1391](#)
- [std::\\_\\_future\\_base::\\_State, 1392](#)
- [std::\\_\\_is\\_location\\_invariant, 1393](#)
- [std::\\_\\_numeric\\_limits\\_base, 1394](#)
  - [digits, 1395](#)
  - [digits10, 1395](#)
  - [has\\_denorm, 1395](#)
  - [has\\_denorm\\_loss, 1396](#)
  - [has\\_infinity, 1396](#)
  - [has\\_quiet\\_NaN, 1396](#)
  - [has\\_signaling\\_NaN, 1396](#)
  - [is\\_bounded, 1396](#)
  - [is\\_exact, 1396](#)
  - [is\\_iec559, 1397](#)
  - [is\\_integer, 1397](#)
  - [is\\_modulo, 1397](#)
  - [is\\_signed, 1397](#)
  - [is\\_specialized, 1397](#)
  - [max\\_digits10, 1397](#)
  - [max\\_exponent, 1397](#)
  - [max\\_exponent10, 1398](#)
  - [min\\_exponent, 1398](#)
  - [min\\_exponent10, 1398](#)

- radix, 1398
- round\_style, 1398
- tinyness\_before, 1398
- traps, 1399
- std::\_\_parallel, 693
- std::\_\_parallel::\_CRandNumber, 1399
- std::\_\_profile, 719
- std::\_\_profile::bitset, 1399
- std::\_\_profile::deque, 1401
- std::\_\_profile::list, 1404
- std::\_\_profile::map, 1406
- std::\_\_profile::multimap, 1409
- std::\_\_profile::multiset, 1411
- std::\_\_profile::set, 1413
- std::\_\_profile::unordered\_map, 1415
- std::\_\_profile::unordered\_multimap, 1416
- std::\_\_profile::unordered\_multiset, 1418
- std::\_\_profile::unordered\_set, 1419
- std::\_Base\_bitset, 1421
- \_M\_w, 1422
- std::\_Base\_bitset< 0 >, 1422
- std::\_Base\_bitset< 1 >, 1424
- std::\_Build\_index\_tuple, 1425
- std::\_Deque\_base, 1426
- \_M\_initialize\_map, 1427
- std::\_Deque\_iterator, 1428
- \_M\_set\_node, 1430
- std::\_Derives\_from\_binary\_function, 1430
- std::\_Derives\_from\_unary\_function, 1431
- std::\_Function\_base, 1431
- std::\_Function\_to\_function\_pointer, 1432
- std::\_Fwd\_list\_base, 1433
- std::\_Fwd\_list\_const\_iterator, 1435
- std::\_Fwd\_list\_iterator, 1436
- std::\_Fwd\_list\_node, 1437
- std::\_Fwd\_list\_node\_base, 1438
- std::\_Has\_result\_type\_helper, 1439
- std::\_Index\_tuple, 1440
- std::\_List\_base, 1440
- std::\_List\_const\_iterator, 1442
- std::\_List\_iterator, 1443
- std::\_List\_node, 1444
- \_M\_data, 1445
- std::\_List\_node\_base, 1445
- std::\_Maybe\_get\_result\_type, 1446
- std::\_Maybe\_unary\_or\_binary\_function, 1447
- std::\_Maybe\_unary\_or\_binary\_-  
    function< \_Res, \_T1 >, 1448
- argument\_type, 1449
- result\_type, 1449
- std::\_Maybe\_unary\_or\_binary\_-  
    function< \_Res, \_T1, \_T2  
    >, 1449
- first\_argument\_type, 1451
- result\_type, 1451
- second\_argument\_type, 1451
- std::\_Maybe\_wrap\_member\_pointer, 1451
- std::\_Maybe\_wrap\_member\_pointer< \_  
    Tp\_Class::\* >, 1452
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_-  
    ArgTypes...) const >, 1453
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_-  
    ArgTypes...) const volatile >, 1454
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_-  
    ArgTypes...) volatile >, 1455
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_-  
    ArgTypes...) >, 1456
- std::\_Mu< \_Arg, false, false >, 1457
- std::\_Mu< \_Arg, false, true >, 1458
- std::\_Mu< \_Arg, true, false >, 1458
- std::\_Mu< reference\_wrapper< \_Tp >, 1459
- false, false >, 1459
- std::\_Placeholder, 1459
- std::\_Reference\_wrapper\_base, 1460
- std::\_Safe\_tuple\_element, 1461
- std::\_Safe\_tuple\_element\_impl, 1461
- std::\_Safe\_tuple\_element\_impl< \_\_i, \_-  
    Tuple, false >, 1462
- std::\_Temporary\_buffer, 1463
- \_Temporary\_buffer, 1464
- begin, 1464
- end, 1464
- requested\_size, 1464
- size, 1465
- std::\_Tuple\_impl< \_Idx >, 1465

- 
- std::\_Tuple\_impl< \_Idx, \_Head, \_-  
Tail...>, 1466
  - std::\_Vector\_base, 1467
  - std::\_Weak\_result\_type, 1468
  - std::\_Weak\_result\_type\_impl, 1469
  - std::\_Weak\_result\_type\_impl< \_-  
Res(\*)(\_ArgTypes...)>, 1470
  - std::\_Weak\_result\_type\_impl< \_-  
Res(&)(\_ArgTypes...)>, 1469
  - std::\_Weak\_result\_type\_impl< \_Res(\_-  
ArgTypes...)>, 1470
  - std::\_Weak\_result\_type\_impl< \_Res(\_-  
Class::\*)(\_ArgTypes...) const  
>, 1471
  - std::\_Weak\_result\_type\_impl< \_Res(\_-  
Class::\*)(\_ArgTypes...) const  
volatile >, 1471
  - std::\_Weak\_result\_type\_impl< \_Res(\_-  
Class::\*)(\_ArgTypes...) volatile  
>, 1472
  - std::\_Weak\_result\_type\_impl< \_Res(\_-  
Class::\*)(\_ArgTypes...)>,  
1473
  - std::add\_lvalue\_reference, 1473
  - std::add\_rvalue\_reference, 1474
  - std::adopt\_lock\_t, 1474
  - std::aligned\_storage, 1475
  - std::allocator, 1475
  - std::allocator< void >, 1477
  - std::atomic, 1478
  - std::atomic< \_Tp \* >, 1478
  - std::atomic< bool >, 1479
  - std::atomic< char >, 1480
  - std::atomic< char16\_t >, 1481
  - std::atomic< char32\_t >, 1482
  - std::atomic< int >, 1483
  - std::atomic< long >, 1484
  - std::atomic< long long >, 1484
  - std::atomic< short >, 1485
  - std::atomic< signed char >, 1486
  - std::atomic< unsigned char >, 1487
  - std::atomic< unsigned int >, 1488
  - std::atomic< unsigned long >, 1489
  - std::atomic< unsigned long long >, 1489
  - std::atomic< unsigned short >, 1490
  - std::atomic< void \* >, 1491
  - std::atomic< wchar\_t >, 1492
  - std::auto\_ptr, 1493
    - ~auto\_ptr, 1495
    - auto\_ptr, 1494, 1495
    - element\_type, 1494
    - get, 1496
    - operator\*, 1496
    - operator->, 1496
    - operator=, 1496, 1497
    - release, 1497
    - reset, 1497
  - std::auto\_ptr\_ref, 1498
  - std::back\_insert\_iterator, 1499
    - back\_insert\_iterator, 1501
    - container\_type, 1500
    - difference\_type, 1500
    - iterator\_category, 1500
    - operator\*, 1501
    - operator++, 1501, 1502
    - operator+=, 1502
    - pointer, 1500
    - reference, 1501
    - value\_type, 1501
  - std::bad\_alloc, 1502
    - what, 1503
  - std::bad\_cast, 1504
    - what, 1504
  - std::bad\_exception, 1505
    - what, 1505
  - std::bad\_function\_call, 1506
    - what, 1506
  - std::bad\_typeid, 1507
    - what, 1508
  - std::basic\_filebuf, 1508
    - ~basic\_filebuf, 1514
    - \_M\_buf, 1534
    - \_M\_buf\_locale, 1534
    - \_M\_buf\_size, 1534
    - \_M\_create\_pback, 1514
    - \_M\_destroy\_pback, 1514
    - \_M\_ext\_buf, 1534
    - \_M\_ext\_buf\_size, 1534
    - \_M\_ext\_next, 1535
    - \_M\_in\_beg, 1535
    - \_M\_in\_cur, 1535
    - \_M\_in\_end, 1536
-

[\\_M\\_mode](#), 1536  
[\\_M\\_out\\_beg](#), 1536  
[\\_M\\_out\\_cur](#), 1537  
[\\_M\\_out\\_end](#), 1537  
[\\_M\\_pback](#), 1537  
[\\_M\\_pback\\_cur\\_save](#), 1537  
[\\_M\\_pback\\_end\\_save](#), 1538  
[\\_M\\_pback\\_init](#), 1538  
[\\_M\\_reading](#), 1538  
[\\_M\\_set\\_buffer](#), 1514  
[\\_\\_streambuf\\_type](#), 1512  
[basic\\_filebuf](#), 1514  
[char\\_type](#), 1512  
[close](#), 1515  
[eback](#), 1515  
[egptr](#), 1516  
[eptr](#), 1516  
[gbump](#), 1516  
[getloc](#), 1517  
[gptr](#), 1517  
[imbue](#), 1517  
[in\\_avail](#), 1518  
[int\\_type](#), 1512  
[is\\_open](#), 1518  
[off\\_type](#), 1513  
[open](#), 1519  
[overflow](#), 1520  
[pbackfail](#), 1521  
[pbase](#), 1521  
[pbump](#), 1522  
[pos\\_type](#), 1513  
[pptr](#), 1522  
[pubimbue](#), 1522  
[pubseekoff](#), 1523  
[pubseekpos](#), 1523  
[pubsetbuf](#), 1523  
[pubsync](#), 1524  
[sbumpc](#), 1524  
[seekoff](#), 1524  
[seekpos](#), 1525  
[setbuf](#), 1525, 1526  
[setg](#), 1526  
[setp](#), 1526  
[sgetc](#), 1527  
[sgetn](#), 1527  
[showmanyc](#), 1528  
[snextc](#), 1528  
[sputbackc](#), 1529  
[sputc](#), 1529  
[sputn](#), 1530  
[stossc](#), 1530  
[sungetc](#), 1530  
[sync](#), 1531  
[traits\\_type](#), 1513  
[uflow](#), 1531  
[underflow](#), 1531  
[xsgetn](#), 1532  
[xspn](#), 1533  
[std::basic\\_fstream](#), 1539  
[~basic\\_fstream](#), 1554  
[\\_M\\_gcount](#), 1599  
[\\_M\\_getloc](#), 1555  
[\\_M\\_write](#), 1555  
[\\_\\_ctype\\_type](#), 1547  
[\\_\\_num\\_get\\_type](#), 1547  
[\\_\\_num\\_put\\_type](#), 1547, 1548  
[adjustfield](#), 1599  
[app](#), 1599  
[ate](#), 1599  
[bad](#), 1555  
[badbit](#), 1600  
[basefield](#), 1600  
[basic\\_fstream](#), 1553, 1554  
[beg](#), 1600  
[binary](#), 1600  
[boolalpha](#), 1601  
[char\\_type](#), 1548  
[clear](#), 1556  
[close](#), 1556  
[copyfmt](#), 1556  
[cur](#), 1601  
[dec](#), 1601  
[end](#), 1601  
[eof](#), 1557  
[eofbit](#), 1601  
[event](#), 1553  
[event\\_callback](#), 1548  
[exceptions](#), 1557  
[fail](#), 1558  
[failbit](#), 1602  
[fill](#), 1559  
[fixed](#), 1602

- flags, 1559, 1560
- floatfield, 1602
- flush, 1560
- fmtflags, 1549
- gcount, 1560
- get, 1561–1564
- getline, 1564
- getloc, 1565
- good, 1566
- goodbit, 1602
- hex, 1603
- ignore, 1566, 1567
- imbue, 1568
- in, 1603
- init, 1568
- int\_type, 1550
- internal, 1603
- iostate, 1550
- is\_open, 1568
- isword, 1569
- left, 1603
- narrow, 1569
- oct, 1603
- off\_type, 1551
- open, 1570
- openmode, 1551
- operator void \*, 1570
- operator<<, 1571–1577
- operator>>, 1578–1584
- out, 1604
- peek, 1584
- pos\_type, 1552
- precision, 1584, 1585
- put, 1585
- putback, 1586
- pword, 1586
- rdbuf, 1587
- rdstate, 1588
- read, 1588
- readsome, 1589
- register\_callback, 1589
- right, 1604
- scientific, 1604
- seekdir, 1552
- seekg, 1590
- seekp, 1591
- setf, 1592
- setstate, 1593
- showbase, 1604
- showpoint, 1604
- showpos, 1604
- skipws, 1605
- sync, 1593
- sync\_with\_stdio, 1594
- tellg, 1594
- tellp, 1595
- tie, 1595
- traits\_type, 1552, 1553
- trunc, 1605
- unget, 1596
- unitbuf, 1605
- unsetf, 1596
- uppercase, 1605
- widen, 1597
- width, 1597
- write, 1598
- xalloc, 1598
- std::basic\_ifstream, 1605
  - ~basic\_ifstream, 1617
  - \_M\_gcount, 1652
  - \_M\_getloc, 1618
  - \_\_ctype\_type, 1612
  - \_\_num\_get\_type, 1612
  - \_\_num\_put\_type, 1612
- adjustfield, 1652
- app, 1653
- ate, 1653
- bad, 1618
- badbit, 1653
- basefield, 1653
- basic\_ifstream, 1617
- beg, 1654
- binary, 1654
- boolalpha, 1654
- char\_type, 1613
- clear, 1618
- close, 1619
- copyfmt, 1619
- cur, 1654
- dec, 1654
- end, 1654
- eof, 1620

eofbit, 1655  
event, 1616  
event\_callback, 1613  
exceptions, 1620  
fail, 1621  
failbit, 1655  
fill, 1621, 1622  
fixed, 1655  
flags, 1622  
floatfield, 1655  
fmtflags, 1613  
gcount, 1623  
get, 1623–1626  
getline, 1626, 1627  
getloc, 1628  
good, 1628  
goodbit, 1655  
hex, 1656  
ignore, 1628–1630  
imbue, 1630  
in, 1656  
init, 1631  
int\_type, 1614  
internal, 1656  
iostate, 1614  
is\_open, 1631  
iword, 1631  
left, 1656  
narrow, 1632  
oct, 1657  
off\_type, 1615  
open, 1632  
openmode, 1615  
operator void \*, 1633  
operator>>, 1633–1640  
out, 1657  
peek, 1640  
pos\_type, 1615  
precision, 1640, 1641  
putback, 1641  
pword, 1642  
rdbuf, 1642  
rdstate, 1643  
read, 1643  
readsome, 1644  
register\_callback, 1645  
right, 1657  
scientific, 1657  
seekdir, 1616  
seekg, 1645, 1646  
setf, 1646, 1647  
setstate, 1647  
showbase, 1657  
showpoint, 1657  
showpos, 1658  
skipws, 1658  
sync, 1648  
sync\_with\_stdio, 1648  
tellg, 1649  
tie, 1649  
traits\_type, 1616  
trunc, 1658  
unget, 1650  
unitbuf, 1658  
unsetf, 1650  
uppercase, 1658  
widen, 1651  
width, 1651  
xalloc, 1652  
std::basic\_ios, 1659  
    ~basic\_ios, 1668  
    \_M\_getloc, 1669  
    \_\_ctype\_type, 1663  
    \_\_num\_get\_type, 1663  
    \_\_num\_put\_type, 1663  
adjustfield, 1683  
app, 1683  
ate, 1683  
bad, 1669  
badbit, 1684  
basefield, 1684  
basic\_ios, 1668  
beg, 1684  
binary, 1684  
boolalpha, 1685  
char\_type, 1663  
clear, 1669  
copyfmt, 1669  
cur, 1685  
dec, 1685  
end, 1685  
eof, 1670

eofbit, 1685  
event, 1668  
event\_callback, 1664  
exceptions, 1670, 1671  
fail, 1671  
failbit, 1686  
fill, 1672  
fixed, 1686  
flags, 1672, 1673  
floatfield, 1686  
fmtflags, 1664  
getloc, 1673  
good, 1673  
goodbit, 1686  
hex, 1687  
imbue, 1674  
in, 1687  
init, 1674  
int\_type, 1665  
internal, 1687  
iostate, 1665  
iword, 1674  
left, 1687  
narrow, 1675  
oct, 1687  
off\_type, 1666  
openmode, 1666  
operator void \*, 1675  
out, 1688  
pos\_type, 1666  
precision, 1676  
pword, 1677  
rdbuf, 1677, 1678  
rdstate, 1678  
register\_callback, 1678  
right, 1688  
scientific, 1688  
seekdir, 1667  
setf, 1679  
setstate, 1680  
showbase, 1688  
showpoint, 1688  
showpos, 1688  
skipws, 1689  
sync\_with\_stdio, 1680  
tie, 1681  
traits\_type, 1667  
trunc, 1689  
unitbuf, 1689  
unsetf, 1681  
uppercase, 1689  
widen, 1682  
width, 1682  
xalloc, 1683  
std::basic\_iostream, 1689  
    ~basic\_iostream, 1704  
    \_M\_gcount, 1748  
    \_M\_getloc, 1704  
    \_M\_write, 1704  
    \_\_ctype\_type, 1697  
    \_\_num\_get\_type, 1697  
    \_\_num\_put\_type, 1697, 1698  
adjustfield, 1748  
app, 1748  
ate, 1748  
bad, 1705  
badbit, 1748  
basefield, 1749  
basic\_iostream, 1704  
beg, 1749  
binary, 1749  
boolalpha, 1749  
char\_type, 1698  
clear, 1705  
copyfmt, 1706  
cur, 1750  
dec, 1750  
end, 1750  
eof, 1706  
eofbit, 1750  
event, 1703  
event\_callback, 1699  
exceptions, 1706, 1707  
fail, 1707  
failbit, 1750  
fill, 1708  
fixed, 1751  
flags, 1709  
floatfield, 1751  
flush, 1709  
fmtflags, 1699  
gcount, 1710



get, 1710–1713  
getline, 1713, 1714  
getloc, 1715  
good, 1715  
goodbit, 1751  
hex, 1751  
ignore, 1715–1717  
imbue, 1717  
in, 1752  
init, 1718  
int\_type, 1700  
internal, 1752  
iostate, 1700  
iword, 1718  
left, 1752  
narrow, 1718  
oct, 1752  
off\_type, 1701  
openmode, 1701  
operator void \*, 1719  
operator<<, 1719–1726  
operator>>, 1726–1732  
out, 1752  
peek, 1733  
pos\_type, 1702  
precision, 1733  
put, 1734  
putback, 1734  
pword, 1735  
rdbuf, 1735, 1736  
rdstate, 1736  
read, 1737  
readsome, 1737  
register\_callback, 1738  
right, 1753  
scientific, 1753  
seekdir, 1702  
seekg, 1738, 1739  
seekp, 1740  
setf, 1741  
setstate, 1741  
showbase, 1753  
showpoint, 1753  
showpos, 1753  
skipws, 1753  
sync, 1742  
sync\_with\_stdio, 1743  
tellg, 1743  
tellp, 1743  
tie, 1744  
traits\_type, 1703  
trunc, 1753  
unget, 1745  
unitbuf, 1754  
unsetf, 1745  
uppercase, 1754  
widen, 1745  
width, 1746  
write, 1747  
xalloc, 1747  
std::basic\_istream, 1754  
    ~basic\_istream, 1765  
    \_M\_gcount, 1799  
    \_M\_getloc, 1766  
    \_\_ctype\_type, 1760  
    \_\_num\_get\_type, 1760  
    \_\_num\_put\_type, 1761  
adjustfield, 1799  
app, 1799  
ate, 1799  
bad, 1766  
badbit, 1800  
basefield, 1800  
basic\_istream, 1765  
beg, 1800  
binary, 1800  
boolalpha, 1800  
char\_type, 1761  
clear, 1766  
copyfmt, 1767  
cur, 1801  
dec, 1801  
end, 1801  
eof, 1767  
eofbit, 1801  
event, 1765  
event\_callback, 1761  
exceptions, 1767, 1768  
fail, 1768  
failbit, 1801  
fill, 1769  
fixed, 1802

- flags, 1770
- floatfield, 1802
- fmtflags, 1762
- gcount, 1770
- get, 1771–1773
- getline, 1774, 1775
- getloc, 1775
- good, 1775
- goodbit, 1802
- hex, 1802
- ignore, 1776, 1777
- imbue, 1777
- in, 1803
- init, 1778
- int\_type, 1762
- internal, 1803
- iostate, 1763
- isword, 1778
- left, 1803
- narrow, 1778
- oct, 1803
- off\_type, 1763
- openmode, 1763
- operator void \*, 1779
- operator>>, 1779–1786
- out, 1803
- peek, 1786
- pos\_type, 1764
- precision, 1786, 1787
- putback, 1787
- pword, 1788
- rdbuf, 1788, 1789
- rdstate, 1789
- read, 1790
- readsome, 1790
- register\_callback, 1791
- right, 1804
- scientific, 1804
- seekdir, 1764
- seekg, 1791, 1792
- setf, 1793
- setstate, 1794
- showbase, 1804
- showpoint, 1804
- showpos, 1804
- skipws, 1804
- sync, 1794
- sync\_with\_stdio, 1795
- tellg, 1795
- tie, 1796
- traits\_type, 1764
- trunc, 1804
- unget, 1796
- unitbuf, 1805
- unsetf, 1797
- uppercase, 1805
- widen, 1797
- width, 1798
- xalloc, 1798
- std::basic\_istream::sentry, 1805
  - operator bool, 1807
  - sentry, 1806
  - traits\_type, 1806
- std::basic\_istream, 1807
  - ~basic\_istream, 1819
  - \_M\_gcount, 1854
  - \_M\_getloc, 1820
  - \_\_ctype\_type, 1814
  - \_\_num\_get\_type, 1814
  - \_\_num\_put\_type, 1814
  - adjustfield, 1854
  - app, 1854
  - ate, 1854
  - bad, 1820
  - badbit, 1855
  - basefield, 1855
  - basic\_istream, 1819
  - beg, 1855
  - binary, 1855
  - boolalpha, 1856
  - char\_type, 1815
  - clear, 1820
  - copyfmt, 1821
  - cur, 1856
  - dec, 1856
  - end, 1856
  - eof, 1821
  - eofbit, 1856
  - event, 1818
  - event\_callback, 1815
  - exceptions, 1822
  - fail, 1823

failbit, 1857  
fill, 1823, 1824  
fixed, 1857  
flags, 1824  
floatfield, 1857  
fmtflags, 1815  
gcount, 1825  
get, 1825–1828  
getline, 1828, 1829  
getloc, 1830  
good, 1830  
goodbit, 1857  
hex, 1858  
ignore, 1830, 1831  
imbue, 1832  
in, 1858  
init, 1832  
int\_type, 1816  
internal, 1858  
iostate, 1816  
iword, 1833  
left, 1858  
narrow, 1833  
oct, 1858  
off\_type, 1817  
openmode, 1817  
operator void \*, 1834  
operator>>, 1834–1840  
out, 1859  
peek, 1841  
pos\_type, 1817  
precision, 1841, 1842  
putback, 1842  
pword, 1843  
rdbuf, 1843  
rdstate, 1844  
read, 1844  
readsome, 1845  
register\_callback, 1846  
right, 1859  
scientific, 1859  
seekdir, 1818  
seekg, 1846, 1847  
setf, 1847, 1848  
setstate, 1848  
showbase, 1859  
showpoint, 1859  
showpos, 1859  
skipws, 1859  
str, 1849  
sync, 1849  
sync\_with\_stdio, 1850  
tellg, 1850  
tie, 1851  
traits\_type, 1818  
trunc, 1860  
unget, 1851  
unitbuf, 1860  
unsetf, 1852  
uppercase, 1860  
widen, 1852  
width, 1853  
xalloc, 1853  
std::basic\_ofstream, 1860  
    ~basic\_ofstream, 1872  
    \_M\_getloc, 1872  
    \_M\_write, 1873  
    \_\_ctype\_type, 1867  
    \_\_num\_get\_type, 1867  
    \_\_num\_put\_type, 1867  
adjustfield, 1898  
app, 1898  
ate, 1898  
bad, 1873  
badbit, 1899  
basefield, 1899  
basic\_ofstream, 1871, 1872  
beg, 1899  
binary, 1899  
boolalpha, 1900  
char\_type, 1867  
clear, 1873  
close, 1874  
copyfmt, 1874  
cur, 1900  
dec, 1900  
end, 1900  
eof, 1875  
eofbit, 1900  
event, 1871  
event\_callback, 1867  
exceptions, 1875

- fail, 1876
- failbit, 1901
- fill, 1876, 1877
- fixed, 1901
- flags, 1877
- floatfield, 1901
- flush, 1878
- fmtflags, 1868
- getloc, 1878
- good, 1878
- goodbit, 1901
- hex, 1902
- imbue, 1879
- in, 1902
- init, 1879
- int\_type, 1869
- internal, 1902
- iostate, 1869
- is\_open, 1879
- isword, 1880
- left, 1902
- narrow, 1880
- oct, 1902
- off\_type, 1869
- open, 1881
- openmode, 1870
- operator void \*, 1881
- operator<<, 1882–1888
- out, 1903
- pos\_type, 1870
- precision, 1889
- put, 1889
- pword, 1890
- rdbuf, 1890
- rdstate, 1891
- register\_callback, 1891
- right, 1903
- scientific, 1903
- seekdir, 1870
- seekp, 1892
- setf, 1893
- setstate, 1893
- showbase, 1903
- showpoint, 1903
- showpos, 1903
- skipws, 1904
- sync\_with\_stdio, 1894
- tellp, 1894
- tie, 1895
- traits\_type, 1871
- trunc, 1904
- unitbuf, 1904
- unsetf, 1896
- uppercase, 1904
- widen, 1896
- width, 1896, 1897
- write, 1897
- xalloc, 1898
- std::basic\_ostream, 1904
  - ~basic\_ostream, 1915
  - \_M\_getloc, 1915
  - \_M\_write, 1916
  - \_\_ctype\_type, 1910
  - \_\_num\_get\_type, 1910
  - \_\_num\_put\_type, 1910
- adjustfield, 1940
- app, 1940
- ate, 1941
- bad, 1916
- badbit, 1941
- basefield, 1941
- basic\_ostream, 1915
- beg, 1941
- binary, 1942
- boolalpha, 1942
- char\_type, 1910
- clear, 1916
- copyfmt, 1917
- cur, 1942
- dec, 1942
- end, 1942
- eof, 1917
- eofbit, 1942
- event, 1915
- event\_callback, 1911
- exceptions, 1918
- fail, 1919
- failbit, 1943
- fill, 1919, 1920
- fixed, 1943
- flags, 1920
- floatfield, 1943

flush, 1921  
fmtflags, 1911  
getloc, 1921  
good, 1921  
goodbit, 1943  
hex, 1944  
imbue, 1922  
in, 1944  
init, 1922  
int\_type, 1912  
internal, 1944  
iostate, 1912  
iword, 1922  
left, 1944  
narrow, 1923  
oct, 1945  
off\_type, 1913  
openmode, 1913  
operator void \*, 1923  
operator<<, 1924–1930  
out, 1945  
pos\_type, 1914  
precision, 1930, 1931  
put, 1931  
pword, 1932  
rdbuf, 1932, 1933  
rdstate, 1933  
register\_callback, 1934  
right, 1945  
scientific, 1945  
seekdir, 1914  
seekp, 1934  
setf, 1935  
setstate, 1936  
showbase, 1945  
showpoint, 1945  
showpos, 1946  
skipws, 1946  
sync\_with\_stdio, 1936  
tellp, 1937  
tie, 1937  
traits\_type, 1914  
trunc, 1946  
unitbuf, 1946  
unsetf, 1938  
uppercase, 1946  
widen, 1938  
width, 1939  
write, 1939  
xalloc, 1940  
std::basic\_ostream::sentry, 1947  
    ~sentry, 1947  
    operator bool, 1948  
    sentry, 1947  
std::basic\_ostringstream, 1948  
    ~basic\_ostringstream, 1960  
    \_M\_getloc, 1960  
    \_M\_write, 1961  
    \_\_ctype\_type, 1955  
    \_\_num\_get\_type, 1955  
    \_\_num\_put\_type, 1955  
    adjustfield, 1986  
    app, 1986  
    ate, 1986  
    bad, 1961  
    badbit, 1986  
    basefield, 1986  
    basic\_ostringstream, 1959  
    beg, 1987  
    binary, 1987  
    boolalpha, 1987  
    char\_type, 1955  
    clear, 1961  
    copyfmt, 1962  
    cur, 1987  
    dec, 1987  
    end, 1988  
    eof, 1962  
    eofbit, 1988  
    event, 1959  
    event\_callback, 1955  
    exceptions, 1963  
    fail, 1964  
    failbit, 1988  
    fill, 1964  
    fixed, 1988  
    flags, 1965  
    floatfield, 1988  
    flush, 1966  
    fmtflags, 1956  
    getloc, 1966  
    good, 1966

goodbit, 1989  
hex, 1989  
imbue, 1967  
in, 1989  
init, 1967  
int\_type, 1957  
internal, 1989  
iostate, 1957  
iword, 1967  
left, 1990  
narrow, 1968  
oct, 1990  
off\_type, 1957  
openmode, 1958  
operator void \*, 1968  
operator<<, 1969–1975  
out, 1990  
pos\_type, 1958  
precision, 1975, 1976  
put, 1976  
pword, 1976  
rdbuf, 1977, 1978  
rdstate, 1978  
register\_callback, 1978  
right, 1990  
scientific, 1990  
seekdir, 1958  
seekp, 1978, 1979  
setf, 1979, 1980  
setstate, 1980  
showbase, 1990  
showpoint, 1991  
showpos, 1991  
skipws, 1991  
str, 1981  
sync\_with\_stdio, 1981  
tellp, 1982  
tie, 1982, 1983  
traits\_type, 1959  
trunc, 1991  
unitbuf, 1991  
unsetf, 1983  
uppercase, 1991  
widen, 1983  
width, 1984  
write, 1984  
xalloc, 1985  
std::basic\_regex, 1992  
    ~basic\_regex, 1997  
assign, 1997–2000  
basic\_regex, 1994–1996  
flags, 2000  
getloc, 2000  
imbue, 2001  
mark\_count, 2001  
operator=, 2001, 2002  
swap, 2002  
std::basic\_streambuf, 2003  
    ~basic\_streambuf, 2009  
    \_M\_buf\_locale, 2025  
    \_M\_in\_beg, 2026  
    \_M\_in\_cur, 2026  
    \_M\_in\_end, 2026  
    \_M\_out\_beg, 2027  
    \_M\_out\_cur, 2027  
    \_M\_out\_end, 2027  
    \_\_streambuf\_type, 2007  
basic\_streambuf, 2009  
char\_type, 2007  
eback, 2010  
egptr, 2010  
epptr, 2010  
gbump, 2011  
getloc, 2011  
gptr, 2011  
imbue, 2012  
in\_avail, 2012  
int\_type, 2007  
off\_type, 2008  
overflow, 2013  
pbackfail, 2013  
pbase, 2014  
pbump, 2014  
pos\_type, 2008  
pptr, 2014  
pubimbue, 2015  
pubseekoff, 2015  
pubseekpos, 2016  
pubsetbuf, 2016  
pubsync, 2016  
sbumpc, 2017  
seekoff, 2017

- seekpos, 2017
- setbuf, 2018
- setg, 2018
- setp, 2019
- sgetc, 2019
- sgetn, 2019
- showmanyc, 2020
- snextc, 2020
- sputbackc, 2021
- sputc, 2021
- sputn, 2022
- stoss, 2022
- sungetc, 2022
- sync, 2023
- traits\_type, 2008
- uflow, 2023
- underflow, 2023
- xsgetn, 2024
- xspn, 2025
- std::basic\_string, 2028
  - ~basic\_string, 2037
  - append, 2037–2040
  - assign, 2040–2043
  - at, 2044
  - back, 2045
  - basic\_string, 2034–2036
  - begin, 2045
  - c\_str, 2045
  - capacity, 2046
  - cbegin, 2046
  - cend, 2046
  - clear, 2046
  - compare, 2047–2050
  - copy, 2050
  - crbegin, 2051
  - crend, 2051
  - data, 2051
  - empty, 2052
  - end, 2052
  - erase, 2052, 2053
  - find, 2054, 2055
  - find\_first\_not\_of, 2056, 2057
  - find\_first\_of, 2058, 2059
  - find\_last\_not\_of, 2060, 2061
  - find\_last\_of, 2062, 2063
  - front, 2063, 2064
  - get\_allocator, 2064
  - insert, 2064–2068
  - length, 2069
  - max\_size, 2069
  - npos, 2086
  - operator+=, 2069, 2070
  - operator=, 2071, 2072
  - push\_back, 2073
  - rbegin, 2073, 2074
  - rend, 2074
  - replace, 2074–2080
  - reserve, 2081
  - resize, 2082
  - rfind, 2082–2084
  - shrink\_to\_fit, 2084
  - size, 2085
  - substr, 2085
  - swap, 2086
- std::basic\_stringbuf, 2086
  - \_M\_buf\_locale, 2110
  - \_M\_in\_beg, 2110
  - \_M\_in\_cur, 2110
  - \_M\_in\_end, 2110
  - \_M\_mode, 2111
  - \_M\_out\_beg, 2111
  - \_M\_out\_cur, 2111
  - \_M\_out\_end, 2112
  - \_\_streambuf\_type, 2090
- basic\_stringbuf, 2092
- char\_type, 2090
- eback, 2092
- egptr, 2093
- epptr, 2093
- gbump, 2093
- getloc, 2094
- gptr, 2094
- imbue, 2094
- in\_avail, 2095
- int\_type, 2090
- off\_type, 2091
- overflow, 2095
- pbackfail, 2096
- pbase, 2097
- pbump, 2097
- pos\_type, 2091
- pptr, 2097

- pubimbue, 2098
- pubseekoff, 2098
- pubseekpos, 2098
- pubsetbuf, 2099
- pubsync, 2099
- sbumpc, 2099
- seekoff, 2100
- seekpos, 2100
- setbuf, 2101
- setg, 2101
- setp, 2102
- sgetc, 2102
- sgetn, 2103
- showmanyc, 2103
- snextc, 2104
- sputbackc, 2104
- sputc, 2104
- sputn, 2105
- stossc, 2105
- str, 2105, 2106
- sungetc, 2106
- sync, 2107
- traits\_type, 2091
- uflow, 2107
- underflow, 2107
- xsgetn, 2108
- xspn, 2109
- std::basic\_stringstream, 2112
  - ~basic\_stringstream, 2128
  - \_M\_gcount, 2172
  - \_M\_getloc, 2128
  - \_M\_write, 2129
  - \_\_ctype\_type, 2121
  - \_\_num\_get\_type, 2121
  - \_\_num\_put\_type, 2121
  - adjustfield, 2172
  - app, 2173
  - ate, 2173
  - bad, 2129
  - badbit, 2173
  - basefield, 2173
  - basic\_stringstream, 2127, 2128
  - beg, 2173
  - binary, 2174
  - boolalpha, 2174
  - char\_type, 2122
  - clear, 2129
  - copyfmt, 2130
  - cur, 2174
  - dec, 2174
  - end, 2174
  - eof, 2130
  - eofbit, 2174
  - event, 2127
  - event\_callback, 2122
  - exceptions, 2131
  - fail, 2132
  - failbit, 2175
  - fill, 2132, 2133
  - fixed, 2175
  - flags, 2133
  - floatfield, 2175
  - flush, 2134
  - fmtflags, 2123
  - gcount, 2134
  - get, 2134–2137
  - getline, 2138, 2139
  - getloc, 2139
  - good, 2139
  - goodbit, 2175
  - hex, 2176
  - ignore, 2140, 2141
  - imbue, 2141
  - in, 2176
  - init, 2142
  - int\_type, 2123, 2124
  - internal, 2176
  - iostate, 2124
  - isword, 2142
  - left, 2176
  - narrow, 2143
  - oct, 2177
  - off\_type, 2124, 2125
  - openmode, 2125
  - operator void \*, 2143
  - operator<<, 2143–2150
  - operator>>, 2150–2156
  - out, 2177
  - peek, 2157
  - pos\_type, 2125, 2126
  - precision, 2157
  - put, 2158



- putback, 2158
- pword, 2159
- rdbuf, 2159, 2160
- rdstate, 2160
- read, 2161
- readsome, 2161
- register\_callback, 2162
- right, 2177
- scientific, 2177
- seekdir, 2126
- seekg, 2162, 2163
- seekp, 2164
- setf, 2165
- setstate, 2165
- showbase, 2177
- showpoint, 2177
- showpos, 2178
- skipws, 2178
- str, 2166
- sync, 2167
- sync\_with\_stdio, 2167
- tellg, 2167
- tellp, 2168
- tie, 2168, 2169
- traits\_type, 2126, 2127
- trunc, 2178
- unget, 2169
- unitbuf, 2178
- unsetf, 2170
- uppercase, 2178
- widen, 2170
- width, 2170, 2171
- write, 2171
- xalloc, 2172
- std::bernoulli\_distribution, 2179
  - bernoulli\_distribution, 2180
  - max, 2180
  - min, 2180
  - operator(), 2180
  - p, 2180
  - param, 2180, 2181
  - reset, 2181
  - result\_type, 2179
- std::bernoulli\_distribution::param\_type, 2181
- std::bidirectional\_iterator\_tag, 2182
- std::binary\_function, 2183
  - first\_argument\_type, 2184
  - result\_type, 2184
  - second\_argument\_type, 2184
- std::binary\_negate, 2185
  - first\_argument\_type, 2185
  - result\_type, 2186
  - second\_argument\_type, 2186
- std::binder1st, 2186
  - argument\_type, 2187
  - result\_type, 2187
- std::binder2nd, 2188
  - argument\_type, 2189
  - result\_type, 2189
- std::binomial\_distribution, 2189
  - max, 2191
  - min, 2191
  - operator<<, 2193
  - operator>>, 2193
  - operator(), 2191
  - operator==, 2193
  - p, 2192
  - param, 2192
  - reset, 2192
  - result\_type, 2191
  - t, 2193
- std::binomial\_distribution::param\_type, 2194
- std::bitset, 2195
  - all, 2201
  - any, 2201
  - bitset, 2200, 2201
  - count, 2202
  - flip, 2202
  - none, 2202
  - operator<<, 2203
  - operator<=<=, 2203
  - operator>>, 2203
  - operator>>=, 2204
  - operator~, 2206
  - operator^=, 2205
  - operator==, 2203
  - operator&=, 2203
  - reset, 2206
  - set, 2206
  - size, 2207

- test, [2207](#)
- to\_string, [2207](#)
- to\_ulong, [2208](#)
- std::bitset::reference, [2208](#)
- std::cauchy\_distribution, [2209](#)
  - max, [2210](#)
  - min, [2210](#)
  - operator(), [2210](#)
  - param, [2211](#)
  - reset, [2211](#)
  - result\_type, [2210](#)
- std::cauchy\_distribution::param\_type, [2212](#)
- std::char\_traits, [2212](#)
- std::char\_traits< \_\_gnu\_cxx::character< V, I, S > >, [2214](#)
- std::char\_traits< char >, [2215](#)
- std::char\_traits< wchar\_t >, [2216](#)
- std::chi\_squared\_distribution, [2217](#)
  - max, [2219](#)
  - min, [2219](#)
  - operator<=, [2220](#)
  - operator>=, [2220](#)
  - operator(), [2219](#)
  - operator==, [2220](#)
  - param, [2219](#)
  - reset, [2220](#)
  - result\_type, [2218](#)
- std::chi\_squared\_distribution::param\_type, [2221](#)
- std::chrono, [726](#)
  - duration\_cast, [730](#)
  - hours, [729](#)
  - microseconds, [729](#)
  - milliseconds, [729](#)
  - minutes, [729](#)
  - nanoseconds, [729](#)
  - seconds, [729](#)
  - time\_point\_cast, [730](#)
- std::chrono::duration, [2222](#)
- std::chrono::duration\_values, [2223](#)
- std::chrono::system\_clock, [2224](#)
- std::chrono::time\_point, [2225](#)
- std::chrono::treat\_as\_floating\_point, [2226](#)
- std::codecvt, [2226](#)
  - do\_out, [2229](#)
  - in, [2229](#)
  - out, [2230](#)
  - unshift, [2231](#)
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, [2232](#)
  - do\_out, [2235](#)
  - in, [2235](#)
  - out, [2236](#)
  - unshift, [2237](#)
- std::codecvt< char, char, mbstate\_t >, [2238](#)
  - do\_out, [2240](#)
  - in, [2240](#)
  - out, [2241](#)
  - unshift, [2242](#)
- std::codecvt< wchar\_t, char, mbstate\_t >, [2243](#)
  - do\_out, [2245](#)
  - in, [2245](#)
  - out, [2246](#)
  - unshift, [2247](#)
- std::codecvt\_base, [2248](#)
- std::codecvt\_byname, [2249](#)
  - do\_out, [2252](#)
  - in, [2252](#)
  - out, [2253](#)
  - unshift, [2254](#)
- std::collate, [2255](#)
  - ~collate, [2259](#)
  - char\_type, [2258](#)
  - collate, [2258](#)
  - compare, [2259](#)
  - do\_compare, [2259](#)
  - do\_hash, [2260](#)
  - do\_transform, [2260](#)
  - hash, [2261](#)
  - id, [2262](#)
  - string\_type, [2258](#)
  - transform, [2261](#)
- std::collate\_byname, [2262](#)
  - char\_type, [2265](#)
  - compare, [2265](#)
  - do\_compare, [2265](#)
  - do\_hash, [2266](#)
  - do\_transform, [2266](#)
  - hash, [2267](#)

- id, 2268
- string\_type, 2265
- transform, 2267
- std::complex, 2268
  - complex, 2270
  - operator+=, 2270
  - operator-=, 2270
  - value\_type, 2270
- std::condition\_variable, 2271
- std::condition\_variable\_any, 2272
- std::conditional, 2273
- std::const\_mem\_fun1\_ref\_t, 2273
  - first\_argument\_type, 2275
  - result\_type, 2275
  - second\_argument\_type, 2275
- std::const\_mem\_fun1\_t, 2275
  - first\_argument\_type, 2277
  - result\_type, 2277
  - second\_argument\_type, 2277
- std::const\_mem\_fun\_ref\_t, 2277
  - argument\_type, 2279
  - result\_type, 2279
- std::const\_mem\_fun\_t, 2279
  - argument\_type, 2281
  - result\_type, 2281
- std::ctype, 2281
  - char\_type, 2284
  - do\_is, 2285
  - do\_narrow, 2286
  - do\_scan\_is, 2287
  - do\_scan\_not, 2287
  - do\_tolower, 2288
  - do\_toupper, 2289
  - do\_widen, 2289, 2290
  - id, 2296
  - is, 2290, 2291
  - narrow, 2291, 2292
  - scan\_is, 2293
  - scan\_not, 2293
  - tolower, 2293, 2294
  - toupper, 2294, 2295
  - widen, 2295, 2296
- std::ctype< char >, 2297
  - ~ctype, 2300
  - char\_type, 2300
  - classic\_table, 2301
  - ctype, 2300
  - do\_narrow, 2301
  - do\_tolower, 2302
  - do\_toupper, 2303
  - do\_widen, 2303, 2304
  - id, 2310
  - is, 2304, 2305
  - narrow, 2305, 2306
  - scan\_is, 2306
  - scan\_not, 2307
  - table, 2307
  - table\_size, 2310
  - tolower, 2307, 2308
  - toupper, 2308, 2309
  - widen, 2309, 2310
- std::ctype< wchar\_t >, 2311
  - ~ctype, 2315
  - char\_type, 2314
  - ctype, 2315
  - do\_is, 2315, 2316
  - do\_narrow, 2317, 2318
  - do\_scan\_is, 2319
  - do\_scan\_not, 2320
  - do\_tolower, 2321, 2322
  - do\_toupper, 2323, 2324
  - do\_widen, 2324, 2325
  - id, 2331
  - is, 2326
  - narrow, 2327
  - scan\_is, 2328
  - scan\_not, 2328
  - tolower, 2329
  - toupper, 2329, 2330
  - widen, 2330, 2331
- std::ctype\_base, 2332
- std::ctype\_byname, 2333
  - char\_type, 2335
  - do\_is, 2336
  - do\_narrow, 2336, 2337
  - do\_scan\_is, 2338
  - do\_scan\_not, 2338
  - do\_tolower, 2339
  - do\_toupper, 2340
  - do\_widen, 2340, 2341
  - id, 2347
  - is, 2341, 2342

- narrow, [2342](#), [2343](#)
- scan\_is, [2344](#)
- scan\_not, [2344](#)
- tolower, [2344](#), [2345](#)
- toupper, [2345](#), [2346](#)
- widen, [2346](#), [2347](#)
- std::ctype\_byname< char >, [2348](#)
- char\_type, [2350](#)
- classic\_table, [2351](#)
- do\_narrow, [2351](#)
- do\_tolower, [2352](#)
- do\_toupper, [2353](#)
- do\_widen, [2353](#), [2354](#)
- id, [2360](#)
- is, [2354](#), [2355](#)
- narrow, [2355](#), [2356](#)
- scan\_is, [2356](#)
- scan\_not, [2357](#)
- table, [2357](#)
- table\_size, [2360](#)
- tolower, [2357](#), [2358](#)
- toupper, [2358](#), [2359](#)
- widen, [2359](#), [2360](#)
- std::decay, [2361](#)
- std::decimal, [730](#)
- decimal32\_to\_long\_long, [741](#)
- std::decimal::decimal128, [2361](#)
- decimal128, [2363](#)
- std::decimal::decimal32, [2363](#)
- decimal32, [2365](#)
- std::decimal::decimal64, [2365](#)
- decimal64, [2367](#)
- std::default\_delete, [2367](#)
- std::defer\_lock\_t, [2368](#)
- std::deque, [2369](#)
- ~deque, [2378](#)
- \_M\_fill\_initialize, [2378](#)
- \_M\_initialize\_map, [2379](#)
- \_M\_new\_elements\_at\_back, [2379](#)
- \_M\_new\_elements\_at\_front, [2379](#)
- \_M\_pop\_back\_aux, [2380](#)
- \_M\_pop\_front\_aux, [2380](#)
- \_M\_push\_back\_aux, [2380](#)
- \_M\_push\_front\_aux, [2380](#)
- \_M\_range\_check, [2381](#)
- \_M\_range\_initialize, [2381](#)
- \_M\_reallocate\_map, [2382](#)
- \_M\_reserve\_elements\_at\_back, [2382](#)
- \_M\_reserve\_elements\_at\_front, [2382](#)
- \_M\_reserve\_map\_at\_back, [2383](#)
- \_M\_reserve\_map\_at\_front, [2383](#)
- assign, [2383](#), [2384](#)
- at, [2384](#), [2385](#)
- back, [2385](#), [2386](#)
- begin, [2386](#)
- cbegin, [2386](#)
- cend, [2386](#)
- clear, [2387](#)
- crbegin, [2387](#)
- crend, [2387](#)
- deque, [2375](#)–[2377](#)
- emplace, [2387](#)
- empty, [2388](#)
- end, [2388](#)
- erase, [2388](#), [2389](#)
- front, [2389](#), [2390](#)
- get\_allocator, [2390](#)
- insert, [2390](#)–[2392](#)
- max\_size, [2392](#)
- operator=, [2392](#), [2393](#)
- pop\_back, [2394](#)
- pop\_front, [2395](#)
- push\_back, [2395](#)
- push\_front, [2395](#)
- rbegin, [2396](#)
- rend, [2396](#)
- resize, [2397](#)
- shrink\_to\_fit, [2397](#)
- size, [2397](#)
- swap, [2398](#)
- std::discard\_block\_engine, [2398](#)
- base, [2402](#)
- discard, [2402](#)
- discard\_block\_engine, [2400](#), [2401](#)
- max, [2402](#)
- min, [2402](#)
- operator<<, [2404](#)
- operator>>, [2404](#)
- operator(), [2402](#)
- operator==, [2404](#)

- result\_type, 2400
- seed, 2403
- std::discrete\_distribution, 2405
  - max, 2407
  - min, 2407
  - operator<<, 2408
  - operator>>, 2408
  - operator(), 2407
  - param, 2407
  - probabilities, 2408
  - reset, 2408
  - result\_type, 2406
- std::discrete\_distribution::param\_type, 2409
- std::divides, 2410
  - first\_argument\_type, 2411
  - result\_type, 2411
  - second\_argument\_type, 2411
- std::domain\_error, 2412
  - what, 2412
- std::enable\_if, 2413
- std::enable\_shared\_from\_this, 2413
- std::equal\_to, 2414
  - first\_argument\_type, 2416
  - result\_type, 2416
  - second\_argument\_type, 2416
- std::error\_category, 2416
- std::error\_code, 2417
- std::error\_condition, 2418
- std::exception, 2418
  - what, 2420
- std::exponential\_distribution, 2420
  - exponential\_distribution, 2422
  - lambda, 2422
  - max, 2422
  - min, 2422
  - operator(), 2422
  - param, 2423
  - reset, 2423
  - result\_type, 2421
- std::exponential\_distribution::param\_type, 2424
- std::extreme\_value\_distribution, 2424
  - a, 2426
  - b, 2426
  - max, 2426
  - min, 2426
  - operator(), 2426
  - param, 2427
  - reset, 2427
  - result\_type, 2425
- std::extreme\_value\_distribution::param\_type, 2427
- std::fisher\_f\_distribution, 2428
  - max, 2430
  - min, 2430
  - operator<<, 2431
  - operator>>, 2431
  - operator(), 2430
  - operator==, 2431
  - param, 2430
  - reset, 2431
  - result\_type, 2429
- std::fisher\_f\_distribution::param\_type, 2432
- std::forward\_iterator\_tag, 2433
- std::forward\_list, 2434
  - ~forward\_list, 2440
  - assign, 2441
  - before\_begin, 2442
  - begin, 2442
  - cbegin, 2443
  - cbegin, 2443
  - cend, 2443
  - clear, 2443
  - emplace\_after, 2444
  - emplace\_front, 2444
  - empty, 2444
  - end, 2445
  - erase\_after, 2445, 2446
  - forward\_list, 2438–2440
  - front, 2446
  - get\_allocator, 2446
  - insert\_after, 2447, 2448
  - max\_size, 2449
  - merge, 2449
  - operator=, 2450
  - pop\_front, 2451
  - push\_front, 2451
  - remove, 2451
  - remove\_if, 2452
  - resize, 2452

- reverse, 2453
- sort, 2453
- splice\_after, 2453, 2454
- swap, 2455
- unique, 2455
- std::fpos, 2456
  - fpos, 2457
  - operator streamoff, 2457
  - operator+, 2457
  - operator+=", 2457
  - operator-, 2457
  - operator=, 2458
  - state, 2458
- std::front\_insert\_iterator, 2458
  - container\_type, 2460
  - difference\_type, 2460
  - front\_insert\_iterator, 2461
  - iterator\_category, 2460
  - operator\*, 2461
  - operator++, 2461
  - operator=, 2462
  - pointer, 2460
  - reference, 2460
  - value\_type, 2461
- std::function< \_Res(\_ArgTypes...)>, 2462
  - function, 2465, 2466
  - operator bool, 2466
  - operator(), 2466
  - operator=, 2467–2469
  - swap, 2469
  - target, 2469
  - target\_type, 2470
- std::future, 2470
  - \_M\_get\_result, 2472
  - future, 2472
  - get, 2472
- std::future< \_Res & >, 2473
  - \_M\_get\_result, 2475
  - future, 2475
  - get, 2475
- std::future< void >, 2475
  - \_M\_get\_result, 2477
  - future, 2477
  - get, 2477
- std::future\_error, 2478
- what, 2479
- std::gamma\_distribution, 2479
  - alpha, 2481
  - beta, 2481
  - gamma\_distribution, 2481
  - max, 2481
  - min, 2481
  - operator<<, 2483
  - operator>>, 2484
  - operator(), 2482
  - operator==, 2483
  - param, 2482
  - reset, 2483
  - result\_type, 2481
- std::gamma\_distribution::param\_type, 2484
- std::geometric\_distribution, 2485
  - max, 2486
  - min, 2486
  - operator(), 2487
  - p, 2487
  - param, 2487
  - reset, 2488
  - result\_type, 2486
- std::geometric\_distribution::param\_type, 2488
- std::greater, 2489
  - first\_argument\_type, 2490
  - result\_type, 2490
  - second\_argument\_type, 2490
- std::greater\_equal, 2490
  - first\_argument\_type, 2492
  - result\_type, 2492
  - second\_argument\_type, 2492
- std::gslice, 2492
- std::gslice\_array, 2493
- std::has\_nothrow\_copy\_assign, 2495
- std::has\_nothrow\_copy\_constructor, 2495
- std::has\_nothrow\_default\_constructor, 2496
- std::has\_trivial\_copy\_assign, 2496
- std::has\_trivial\_copy\_constructor, 2497
- std::has\_trivial\_default\_constructor, 2497
- std::has\_trivial\_destructor, 2497
- std::hash, 2498
  - argument\_type, 2500

- result\_type, 2500
- std::hash< \_\_debug::bitset< \_Nb > >, 2500
  - argument\_type, 2502
  - result\_type, 2502
- std::hash< \_\_debug::vector< bool, \_Alloc > >, 2502
  - argument\_type, 2504
  - result\_type, 2504
- std::hash< \_\_gnu\_cxx::throw\_value\_-limit >, 2504
  - argument\_type, 2506
  - result\_type, 2506
- std::hash< \_\_gnu\_cxx::throw\_value\_-random >, 2506
  - argument\_type, 2508
  - result\_type, 2508
- std::hash< \_\_profile::bitset< \_Nb > >, 2508
  - argument\_type, 2510
  - result\_type, 2510
- std::hash< \_\_profile::vector< bool, \_Alloc > >, 2510
  - argument\_type, 2512
  - result\_type, 2512
- std::hash< \_Tp \* >, 2512
  - argument\_type, 2514
  - result\_type, 2514
- std::hash< error\_code >, 2514
  - argument\_type, 2516
  - result\_type, 2516
- std::hash< shared\_ptr< \_Tp > >, 2516
  - argument\_type, 2518
  - result\_type, 2518
- std::hash< string >, 2518
  - argument\_type, 2520
  - result\_type, 2520
- std::hash< thread::id >, 2520
  - argument\_type, 2522
  - result\_type, 2522
- std::hash< ul6string >, 2522
  - argument\_type, 2524
  - result\_type, 2524
- std::hash< u32string >, 2524
  - argument\_type, 2526
  - result\_type, 2526
- std::hash< unique\_ptr< \_Tp, \_Tp\_-Deleter > >, 2526
  - argument\_type, 2528
  - result\_type, 2528
- std::hash< wstring >, 2528
  - argument\_type, 2530
  - result\_type, 2530
- std::hash<::bitset< \_Nb > >, 2530
  - argument\_type, 2532
  - result\_type, 2532
- std::hash<::vector< bool, \_Alloc > >, 2532
  - argument\_type, 2534
  - result\_type, 2534
- std::identity, 2534
- std::independent\_bits\_engine, 2535
  - base, 2538
  - discard, 2538
  - independent\_bits\_engine, 2536, 2537
  - max, 2538
  - min, 2538
  - operator>>, 2540
  - operator(), 2539
  - operator==, 2540
  - result\_type, 2536
  - seed, 2539
- std::indirect\_array, 2541
  - operator<=, 2543
  - operator>=, 2543
  - operator\*=, 2543
  - operator^=, 2543
  - operator+=, 2543
  - operator-=, 2543
  - operator/=: 2543
  - operator%=: 2542
  - operator&=: 2542
- std::initializer\_list, 2544
- std::input\_iterator\_tag, 2545
- std::insert\_iterator, 2546
  - container\_type, 2547
  - difference\_type, 2547
  - insert\_iterator, 2548
  - iterator\_category, 2547
  - operator\*, 2548
  - operator++, 2548, 2549

- operator=, 2549
- pointer, 2547
- reference, 2548
- value\_type, 2548
- std::invalid\_argument, 2550
  - what, 2550
- std::ios\_base, 2551
  - ~ios\_base, 2557
  - \_M\_getloc, 2557
  - adjustfield, 2563
  - app, 2563
  - ate, 2563
  - badbit, 2564
  - basefield, 2564
  - beg, 2564
  - binary, 2564
  - boolalpha, 2565
  - cur, 2565
  - dec, 2565
  - end, 2565
  - eofbit, 2565
  - event, 2557
  - event\_callback, 2554
  - failbit, 2566
  - fixed, 2566
  - flags, 2557, 2558
  - floatfield, 2566
  - fmtflags, 2554
  - getloc, 2558
  - goodbit, 2566
  - hex, 2567
  - imbue, 2558
  - in, 2567
  - internal, 2567
  - iostate, 2555
  - isword, 2559
  - left, 2567
  - oct, 2567
  - openmode, 2556
  - out, 2568
  - precision, 2559, 2560
  - pwd, 2560
  - register\_callback, 2560
  - right, 2568
  - scientific, 2568
  - seekdir, 2556
  - setf, 2561
  - showbase, 2568
  - showpoint, 2568
  - showpos, 2568
  - skipws, 2569
  - sync\_with\_stdio, 2561
  - trunc, 2569
  - unitbuf, 2569
  - unsetf, 2562
  - uppercase, 2569
  - width, 2562
  - xalloc, 2563
- std::ios\_base::failure, 2569
  - what, 2570
- std::is\_base\_of, 2571
- std::is\_bind\_expression, 2571
- std::is\_bind\_expression< \_Bind< \_-  
Signature > >, 2572
- std::is\_bind\_expression< \_Bind\_result<  
\_Result, \_Signature > >, 2572
- std::is\_constructible, 2573
- std::is\_convertible, 2574
- std::is\_error\_code\_enum, 2574
- std::is\_error\_condition\_enum, 2575
- std::is\_explicitly\_convertible, 2575
- std::is\_lvalue\_reference, 2576
- std::is\_nothrow\_constructible, 2577
- std::is\_placeholder, 2577
- std::is\_placeholder< \_Placeholder< \_-  
Num > >, 2578
- std::is\_pod, 2578
- std::is\_reference, 2579
- std::is\_rvalue\_reference, 2579
- std::is\_signed, 2580
- std::is\_standard\_layout, 2580
- std::is\_trivial, 2580
- std::is\_unsigned, 2581
- std::istream\_iterator, 2581
  - difference\_type, 2583
  - istream\_iterator, 2584
  - iterator\_category, 2583
  - pointer, 2583
  - reference, 2583
  - value\_type, 2583
- std::istreambuf\_iterator, 2584
  - char\_type, 2586



- difference\_type, 2586
- equal, 2588
- int\_type, 2586
- istream\_type, 2586
- istreambuf\_iterator, 2587, 2588
- iterator\_category, 2586
- operator\*, 2588
- operator++, 2588
- pointer, 2586
- reference, 2587
- streambuf\_type, 2587
- traits\_type, 2587
- value\_type, 2587
- std::iterator, 2589
  - difference\_type, 2590
  - iterator\_category, 2590
  - pointer, 2590
  - reference, 2590
  - value\_type, 2591
- std::iterator\_traits, 2591
- std::iterator\_traits< \_Tp \* >, 2593
- std::iterator\_traits< const \_Tp \* >, 2593
- std::length\_error, 2594
  - what, 2595
- std::less, 2595
  - first\_argument\_type, 2597
  - result\_type, 2597
  - second\_argument\_type, 2597
- std::less\_equal, 2597
  - first\_argument\_type, 2599
  - result\_type, 2599
  - second\_argument\_type, 2599
- std::linear\_congruential\_engine, 2599
  - discard, 2602
  - increment, 2605
  - linear\_congruential\_engine, 2601
  - max, 2602
  - min, 2602
  - modulus, 2605
  - multiplier, 2605
  - operator<<, 2604
  - operator>>, 2604
  - operator(), 2602
  - operator==, 2604
  - result\_type, 2601
  - seed, 2603
- std::list, 2606
  - \_M\_create\_node, 2613
  - assign, 2613, 2614
  - back, 2614, 2615
  - begin, 2615
  - cbegin, 2615
  - cend, 2615
  - clear, 2615
  - crbegin, 2616
  - crend, 2616
  - emplace, 2616
  - empty, 2617
  - end, 2617
  - erase, 2617, 2618
  - front, 2618
  - get\_allocator, 2619
  - insert, 2619–2621
  - list, 2610–2612
  - max\_size, 2621
  - merge, 2621, 2622
  - operator=, 2622, 2623
  - pop\_back, 2623
  - pop\_front, 2623
  - push\_back, 2624
  - push\_front, 2624
  - rbegin, 2624
  - remove, 2625
  - remove\_if, 2625
  - rend, 2625, 2626
  - resize, 2626
  - reverse, 2626
  - size, 2627
  - sort, 2627
  - splice, 2627, 2628
  - swap, 2629
  - unique, 2629
- std::locale, 2630
  - ~locale, 2634
  - all, 2638
  - category, 2632
  - classic, 2634
  - collate, 2638
  - combine, 2634
  - ctype, 2638
  - global, 2635
  - has\_facet, 2637

- locale, 2632–2634
- messages, 2638
- monetary, 2638
- name, 2635
- none, 2639
- numeric, 2639
- operator(), 2636
- operator=, 2636
- operator==, 2636
- time, 2639
- use\_facet, 2637
- std::locale::facet, 2640
  - ~facet, 2641
  - facet, 2641
- std::locale::id, 2642
  - has\_facet, 2643
  - id, 2642
  - use\_facet, 2643
- std::lock\_guard, 2644
- std::logic\_error, 2644
  - logic\_error, 2645
  - what, 2646
- std::logical\_and, 2646
  - first\_argument\_type, 2647
  - result\_type, 2647
  - second\_argument\_type, 2647
- std::logical\_not, 2648
  - argument\_type, 2649
  - result\_type, 2649
- std::logical\_or, 2649
  - first\_argument\_type, 2651
  - result\_type, 2651
  - second\_argument\_type, 2651
- std::lognormal\_distribution, 2651
  - max, 2653
  - min, 2653
  - operator<<, 2654
  - operator>>, 2655
  - operator(), 2653
  - operator==, 2654
  - param, 2653, 2654
  - reset, 2654
  - result\_type, 2653
- std::lognormal\_distribution::param\_type, 2655
- std::make\_signed, 2656
- std::make\_unsigned, 2656
- std::map, 2657
  - at, 2662
  - begin, 2663
  - cbegin, 2663
  - cend, 2663
  - clear, 2663
  - count, 2664
  - crbegin, 2664
  - crend, 2664
  - empty, 2665
  - end, 2665
  - equal\_range, 2665, 2666
  - erase, 2667
  - find, 2668
  - get\_allocator, 2669
  - insert, 2669–2671
  - key\_comp, 2671
  - lower\_bound, 2671, 2672
  - map, 2660–2662
  - max\_size, 2672
  - operator=, 2672, 2673
  - rbegin, 2674
  - rend, 2674, 2675
  - size, 2675
  - swap, 2675
  - upper\_bound, 2676
  - value\_comp, 2676
- std::mask\_array, 2677
  - operator<=, 2679
  - operator>=, 2679
  - operator\*=, 2679
  - operator^=, 2679
  - operator+=, 2679
  - operator-=, 2679
  - operator/=, 2679
  - operator%=, 2678
  - operator&=, 2678
- std::match\_results, 2680
  - ~match\_results, 2685
  - begin, 2685
  - cbegin, 2685
  - cend, 2685
  - empty, 2686
  - end, 2686
  - format, 2686, 2687

- get\_allocator, 2687
- length, 2687
- match\_results, 2684
- max\_size, 2688
- operator=, 2688
- position, 2688
- prefix, 2689
- size, 2689
- str, 2690
- suffix, 2690
- swap, 2690
- std::mem\_fun1\_ref\_t, 2691
  - first\_argument\_type, 2692
  - result\_type, 2692
  - second\_argument\_type, 2692
- std::mem\_fun1\_t, 2692
  - first\_argument\_type, 2694
  - result\_type, 2694
  - second\_argument\_type, 2694
- std::mem\_fun\_ref\_t, 2694
  - argument\_type, 2696
  - result\_type, 2696
- std::mem\_fun\_t, 2696
  - argument\_type, 2698
  - result\_type, 2698
- std::messages, 2698
  - ~messages, 2702
  - char\_type, 2701
  - id, 2702
  - messages, 2701
  - string\_type, 2701
- std::messages\_base, 2702
- std::messages\_byname, 2703
  - char\_type, 2706
  - id, 2706
  - string\_type, 2706
- std::minus, 2706
  - first\_argument\_type, 2708
  - result\_type, 2708
  - second\_argument\_type, 2708
- std::modulus, 2708
  - first\_argument\_type, 2710
  - result\_type, 2710
  - second\_argument\_type, 2710
- std::money\_base, 2710
- std::money\_get, 2712
  - ~money\_get, 2715
- char\_type, 2714
- do\_get, 2715
- get, 2716, 2717
- id, 2717
- iter\_type, 2714
- money\_get, 2715
- string\_type, 2714
- std::money\_put, 2718
  - ~money\_put, 2720
- char\_type, 2720
- do\_put, 2721
- id, 2723
- iter\_type, 2720
- money\_put, 2720
- put, 2722, 2723
- string\_type, 2720
- std::moneypunct, 2724
  - ~moneypunct, 2728
- char\_type, 2726
- curr\_symbol, 2728
- decimal\_point, 2728
- do\_curr\_symbol, 2728
- do\_decimal\_point, 2729
- do\_frac\_digits, 2729
- do\_grouping, 2730
- do\_neg\_format, 2730
- do\_negative\_sign, 2730
- do\_pos\_format, 2731
- do\_positive\_sign, 2731
- do\_thousands\_sep, 2732
- frac\_digits, 2732
- grouping, 2732
- id, 2736
- intl, 2736
- moneypunct, 2727
- neg\_format, 2733
- negative\_sign, 2734
- pos\_format, 2734
- positive\_sign, 2735
- string\_type, 2726
- thousands\_sep, 2735
- std::moneypunct\_byname, 2736
  - char\_type, 2739
  - curr\_symbol, 2740
  - decimal\_point, 2740

- do\_curr\_symbol, 2740
- do\_decimal\_point, 2741
- do\_frac\_digits, 2741
- do\_grouping, 2741
- do\_neg\_format, 2742
- do\_negative\_sign, 2742
- do\_pos\_format, 2743
- do\_positive\_sign, 2743
- do\_thousands\_sep, 2744
- frac\_digits, 2744
- grouping, 2744
- id, 2748
- intl, 2748
- neg\_format, 2745
- negative\_sign, 2746
- pos\_format, 2746
- positive\_sign, 2747
- string\_type, 2739
- thousands\_sep, 2747
- std::move\_iterator, 2748
- std::multimap, 2749
  - begin, 2755
  - cbegin, 2755
  - cend, 2755
  - clear, 2755
  - count, 2756
  - crbegin, 2756
  - crend, 2756
  - empty, 2756
  - end, 2757
  - equal\_range, 2757, 2758
  - erase, 2758, 2759
  - find, 2760
  - get\_allocator, 2760
  - insert, 2761, 2762
  - key\_comp, 2762
  - lower\_bound, 2763
  - max\_size, 2764
  - multimap, 2752–2754
  - operator=, 2764, 2765
  - rbegin, 2765
  - rend, 2765, 2766
  - size, 2766
  - swap, 2766
  - upper\_bound, 2767
  - value\_comp, 2767
- std::multiplies, 2768
  - first\_argument\_type, 2769
  - result\_type, 2769
  - second\_argument\_type, 2769
- std::multiset, 2769
  - begin, 2774
  - cbegin, 2774
  - cend, 2775
  - clear, 2775
  - count, 2775
  - crbegin, 2775
  - crend, 2776
  - empty, 2776
  - end, 2776
  - equal\_range, 2776, 2777
  - erase, 2777, 2778
  - find, 2779
  - get\_allocator, 2780
  - insert, 2780, 2781
  - key\_comp, 2782
  - lower\_bound, 2782
  - max\_size, 2783
  - multiset, 2772–2774
  - operator<, 2786
  - operator=, 2783, 2784
  - operator==, 2786
  - rbegin, 2784
  - rend, 2784
  - size, 2784
  - swap, 2785
  - upper\_bound, 2785
  - value\_comp, 2786
- std::mutex, 2787
- std::negate, 2788
  - argument\_type, 2789
  - result\_type, 2789
- std::negative\_binomial\_distribution, 2789
  - k, 2791
  - max, 2791
  - min, 2791
  - operator<<, 2793
  - operator>>, 2793
  - operator(), 2791
  - operator==, 2793
  - p, 2791
  - param, 2792

- reset, 2792
- result\_type, 2791
- std::negative\_binomial\_-
  - distribution::param\_type, 2794
- std::nested\_exception, 2794
- std::normal\_distribution, 2795
  - max, 2797
  - mean, 2797
  - min, 2797
  - normal\_distribution, 2797
  - operator<<, 2799
  - operator>>, 2800
  - operator(), 2798
  - operator==, 2800
  - param, 2798
  - reset, 2799
  - result\_type, 2797
  - stddev, 2799
- std::normal\_distribution::param\_type, 2800
- std::not\_equal\_to, 2801
  - first\_argument\_type, 2803
  - result\_type, 2803
  - second\_argument\_type, 2803
- std::num\_get, 2803
  - ~num\_get, 2807
  - char\_type, 2806
  - do\_get, 2807–2813
  - get, 2814–2822
  - id, 2823
  - iter\_type, 2806
  - num\_get, 2807
- std::num\_put, 2823
  - ~num\_put, 2827
  - char\_type, 2826
  - do\_put, 2827–2830
  - id, 2838
  - iter\_type, 2826
  - num\_put, 2827
  - put, 2831–2837
- std::numeric\_limits, 2838
  - denorm\_min, 2840
  - digits, 2841
  - digits10, 2842
  - epsilon, 2840
  - has\_denorm, 2842
  - has\_denorm\_loss, 2842
  - has\_infinity, 2842
  - has\_quiet\_NaN, 2842
  - has\_signaling\_NaN, 2842
  - infinity, 2840
  - is\_bounded, 2842
  - is\_exact, 2843
  - is\_iec559, 2843
  - is\_integer, 2843
  - is\_modulo, 2843
  - is\_signed, 2843
  - is\_specialized, 2844
  - lowest, 2840
  - max, 2840
  - max\_digits10, 2844
  - max\_exponent, 2844
  - max\_exponent10, 2844
  - min, 2841
  - min\_exponent, 2844
  - min\_exponent10, 2844
  - quiet\_NaN, 2841
  - radix, 2845
  - round\_error, 2841
  - round\_style, 2845
  - signaling\_NaN, 2841
  - tinyness\_before, 2845
  - traps, 2845
- std::numeric\_limits< bool >, 2846
- std::numeric\_limits< char >, 2847
- std::numeric\_limits< char16\_t >, 2848
- std::numeric\_limits< char32\_t >, 2849
- std::numeric\_limits< double >, 2851
- std::numeric\_limits< float >, 2852
- std::numeric\_limits< int >, 2853
- std::numeric\_limits< long >, 2855
- std::numeric\_limits< long double >, 2856
- std::numeric\_limits< long long >, 2857
- std::numeric\_limits< short >, 2859
- std::numeric\_limits< signed char >, 2860
- std::numeric\_limits< unsigned char >, 2861
- std::numeric\_limits< unsigned int >, 2863

- std::numeric\_limits< unsigned long >, 2864
- std::numeric\_limits< unsigned long long >, 2865
- std::numeric\_limits< unsigned short >, 2867
- std::numeric\_limits< wchar\_t >, 2868
- std::numpunct, 2869
  - ~numpunct, 2873
  - char\_type, 2872
  - decimal\_point, 2873
  - do\_decimal\_point, 2874
  - do\_falsename, 2874
  - do\_grouping, 2874
  - do\_thousands\_sep, 2875
  - do\_truename, 2875
  - falsename, 2876
  - grouping, 2876
  - id, 2877
  - numpunct, 2872, 2873
  - string\_type, 2872
  - thousands\_sep, 2876
  - truename, 2877
- std::numpunct\_byname, 2878
  - char\_type, 2880
  - decimal\_point, 2880
  - do\_decimal\_point, 2880
  - do\_falsename, 2881
  - do\_grouping, 2881
  - do\_thousands\_sep, 2881
  - do\_truename, 2882
  - falsename, 2882
  - grouping, 2882
  - id, 2884
  - string\_type, 2880
  - thousands\_sep, 2883
  - truename, 2883
- std::once\_flag, 2884
  - call\_once, 2885
- std::ostream\_iterator, 2885
  - char\_type, 2886
  - difference\_type, 2886
  - iterator\_category, 2887
  - operator=, 2889
  - ostream\_iterator, 2888
  - ostream\_type, 2887
  - pointer, 2887
  - reference, 2887
  - traits\_type, 2887
  - value\_type, 2887
- std::ostreambuf\_iterator, 2889
  - char\_type, 2891
  - difference\_type, 2891
  - failed, 2893
  - iterator\_category, 2891
  - operator\*, 2893
  - operator++, 2893
  - operator=, 2893
  - ostream\_type, 2891
  - ostreambuf\_iterator, 2892
  - pointer, 2891
  - reference, 2891
  - streambuf\_type, 2891
  - traits\_type, 2892
  - value\_type, 2892
- std::out\_of\_range, 2894
  - what, 2894
- std::output\_iterator\_tag, 2895
- std::overflow\_error, 2896
  - what, 2896
- std::owner\_less< shared\_ptr< \_Tp > >, 2897
- std::owner\_less< weak\_ptr< \_Tp > >, 2897
- std::packaged\_task< \_Res(\_ArgTypes...)>, 2898
- std::pair, 2899
  - first, 2901
  - first\_type, 2900
  - pair, 2900, 2901
  - second, 2901
  - second\_type, 2900
- std::piecewise\_constant\_distribution, 2901
  - densities, 2903
  - intervals, 2903
  - max, 2903
  - min, 2903
  - operator<<, 2905
  - operator>>, 2905
  - operator(), 2904
  - param, 2904

- reset, 2904
- result\_type, 2903
- std::piecewise\_constant\_  
distribution::param\_type,  
2906
- std::piecewise\_linear\_distribution, 2907
  - densities, 2908
  - intervals, 2908
  - max, 2909
  - min, 2909
  - operator<<, 2910
  - operator>>, 2910
  - operator(), 2909
  - param, 2909
  - reset, 2910
  - result\_type, 2908
- std::piecewise\_linear\_  
distribution::param\_type,  
2911
- std::placeholders, 741
- std::plus, 2912
  - first\_argument\_type, 2913
  - result\_type, 2913
  - second\_argument\_type, 2913
- std::pointer\_to\_binary\_function, 2914
  - first\_argument\_type, 2915
  - result\_type, 2915
  - second\_argument\_type, 2915
- std::pointer\_to\_unary\_function, 2915
  - argument\_type, 2917
  - result\_type, 2917
- std::poisson\_distribution, 2917
  - max, 2919
  - mean, 2919
  - min, 2919
  - operator<<, 2921
  - operator>>, 2921
  - operator(), 2919
  - operator==, 2921
  - param, 2920
  - reset, 2920
  - result\_type, 2918
- std::poisson\_distribution::param\_type,  
2922
- std::priority\_queue, 2922
  - empty, 2925
  - pop, 2925
  - priority\_queue, 2924
  - push, 2926
  - size, 2926
  - top, 2926
- std::promise, 2927
- std::promise< \_Res & >, 2928
- std::promise< void >, 2928
- std::queue, 2929
  - back, 2931
  - c, 2933
  - empty, 2931
  - front, 2931, 2932
  - pop, 2932
  - push, 2932
  - queue, 2931
  - size, 2932
- std::random\_access\_iterator\_tag, 2933
- std::random\_device, 2934
  - result\_type, 2935
- std::range\_error, 2936
  - what, 2936
- std::ratio, 2937
- std::ratio\_add, 2938
- std::ratio\_divide, 2938
- std::ratio\_equal, 2939
- std::ratio\_greater, 2939
- std::ratio\_greater\_equal, 2940
- std::ratio\_less, 2940
- std::ratio\_less\_equal, 2941
- std::ratio\_multiply, 2941
- std::ratio\_not\_equal, 2942
- std::ratio\_subtract, 2942
- std::raw\_storage\_iterator, 2943
  - difference\_type, 2944
  - iterator\_category, 2944
  - pointer, 2944
  - reference, 2944
  - value\_type, 2944
- std::recursive\_mutex, 2945
- std::recursive\_timed\_mutex, 2945
- std::reference\_wrapper, 2946
- std::regex\_constants, 741
  - \_\_match\_flag, 744
  - \_\_syntax\_option, 744
- awk, 746

- basic, 746
- collate, 746
- ECMAScript, 747
- egrep, 747
- error\_backref, 744
- error\_badbrace, 744
- error\_badrepeat, 745
- error\_brace, 745
- error\_brack, 745
- error\_collate, 745
- error\_complexity, 745
- error\_ctype, 745
- error\_escape, 745
- error\_paren, 745
- error\_range, 746
- error\_space, 746
- error\_stack, 746
- error\_type, 744
- extended, 747
- format\_default, 747
- format\_first\_only, 748
- format\_no\_copy, 748
- format\_sed, 748
- grep, 748
- icase, 749
- match\_any, 749
- match\_continuous, 749
- match\_default, 749
- match\_flag\_type, 743
- match\_not BOL, 749
- match\_not\_bow, 749
- match\_not\_eol, 750
- match\_not\_eow, 750
- match\_not\_null, 750
- match\_prev\_avail, 750
- nosubs, 750
- optimize, 750
- syntax\_option\_type, 743
- std::regex\_error, 2948
  - code, 2949
  - regex\_error, 2949
  - what, 2949
- std::regex\_iterator, 2950
  - operator\*, 2952
  - operator++, 2952
  - operator->, 2953
  - operator=, 2953
  - operator==, 2953
  - regex\_iterator, 2951
- std::regex\_token\_iterator, 2954
  - operator\*, 2958
  - operator++, 2958
  - operator->, 2959
  - operator=, 2959
  - operator==, 2959
  - regex\_token\_iterator, 2955–2957
- std::regex\_traits, 2960
  - getloc, 2961
  - imbue, 2961
  - length, 2962
  - lookup\_classname, 2962
  - lookup\_collatename, 2963
  - regex\_traits, 2961
  - transform, 2964
  - transform\_primary, 2965
  - translate, 2965
  - translate\_nocase, 2965
- std::rel\_ops, 751
  - operator<=, 752
  - operator>, 752
  - operator>=, 752
- std::remove\_reference, 2966
- std::reverse\_iterator, 2967
  - base, 2970
  - difference\_type, 2968
  - iterator\_category, 2968
  - operator\*, 2970
  - operator+, 2971
  - operator++, 2971
  - operator+=, 2971
  - operator-, 2972
  - operator->, 2973
  - operator--, 2972
  - operator-=, 2973
  - pointer, 2969
  - reference, 2969
  - reverse\_iterator, 2969, 2970
  - value\_type, 2969
- std::runtime\_error, 2974
  - runtime\_error, 2975
  - what, 2975
- std::seed\_seq, 2975



- result\_type, 2976
- seed\_seq, 2976
- std::set, 2976
  - allocator\_type, 2979
  - begin, 2985
  - cbegin, 2985
  - cend, 2985
  - clear, 2985
  - const\_iterator, 2979
  - const\_pointer, 2980
  - const\_reference, 2980
  - const\_reverse\_iterator, 2980
  - count, 2986
  - crbegin, 2986
  - crend, 2986
  - difference\_type, 2980
  - empty, 2986
  - end, 2987
  - equal\_range, 2987
  - erase, 2988, 2989
  - find, 2989, 2990
  - get\_allocator, 2990
  - insert, 2990–2992
  - iterator, 2980
  - key\_comp, 2992
  - key\_compare, 2981
  - key\_type, 2981
  - lower\_bound, 2992, 2993
  - max\_size, 2993
  - operator=, 2993, 2994
  - pointer, 2981
  - rbegin, 2994
  - reference, 2981
  - rend, 2995
  - reverse\_iterator, 2981
  - set, 2982–2984
  - size, 2995
  - size\_type, 2982
  - swap, 2995
  - upper\_bound, 2995, 2996
  - value\_comp, 2996
  - value\_compare, 2982
  - value\_type, 2982
- std::shared\_future, 2997
  - \_M\_get\_result, 2999
  - get, 2999
- shared\_future, 2998
- std::shared\_future< \_Res & >, 2999
  - \_M\_get\_result, 3002
  - get, 3002
  - shared\_future, 3001
- std::shared\_future< void >, 3002
  - \_M\_get\_result, 3005
  - shared\_future, 3004
- std::shared\_ptr, 3005
  - allocate\_shared, 3012
  - shared\_ptr, 3007–3011
- std::shuffle\_order\_engine, 3012
  - base, 3016
  - discard, 3016
  - max, 3016
  - min, 3016
  - operator<<, 3018
  - operator>>, 3018
  - operator(), 3016
  - operator==, 3018
  - result\_type, 3014
  - seed, 3017
  - shuffle\_order\_engine, 3014, 3015
- std::slice, 3019
- std::slice\_array, 3020
  - operator<=, 3022
  - operator>=, 3022
  - operator\*=, 3022
  - operator^=, 3022
  - operator+=, 3022
  - operator-=, 3022
  - operator/=: 3022
  - operator%=: 3021
  - operator&=: 3021
- std::stack, 3023
  - empty, 3025
  - pop, 3025
  - push, 3025
  - size, 3026
  - stack, 3025
  - top, 3026
- std::student\_t\_distribution, 3026
  - max, 3028
  - min, 3028
  - operator<<, 3029
  - operator>>, 3030

- operator(), 3028
- operator==, 3029
- param, 3028
- reset, 3029
- result\_type, 3028
- std::student\_t\_distribution::param\_type, 3030
- std::sub\_match, 3032
  - compare, 3034
  - first, 3036
  - first\_type, 3033
  - length, 3035
  - operator string\_type, 3035
  - second, 3036
  - second\_type, 3033
  - str, 3035
- std::system\_error, 3036
  - what, 3037
- std::this\_thread, 753
  - get\_id, 753
  - sleep\_for, 753
  - sleep\_until, 754
  - yield, 754
- std::thread, 3038
  - native\_handle, 3039
- std::thread::id, 3039
- std::time\_base, 3040
- std::time\_get, 3041
  - ~time\_get, 3044
  - char\_type, 3043
  - date\_order, 3044
  - do\_date\_order, 3045
  - do\_get\_date, 3045
  - do\_get\_monthname, 3046
  - do\_get\_time, 3046
  - do\_get\_weekday, 3047
  - do\_get\_year, 3048
  - get\_date, 3048
  - get\_monthname, 3049
  - get\_time, 3050
  - get\_weekday, 3050
  - get\_year, 3051
  - id, 3052
  - iter\_type, 3044
  - time\_get, 3044
- std::time\_get\_byname, 3052
  - char\_type, 3055
  - date\_order, 3055
  - do\_date\_order, 3056
  - do\_get\_date, 3056
  - do\_get\_monthname, 3057
  - do\_get\_time, 3057
  - do\_get\_weekday, 3058
  - do\_get\_year, 3059
  - get\_date, 3059
  - get\_monthname, 3060
  - get\_time, 3061
  - get\_weekday, 3061
  - get\_year, 3062
  - id, 3063
  - iter\_type, 3055
- std::time\_put, 3063
  - ~time\_put, 3066
  - char\_type, 3065
  - do\_put, 3066
  - id, 3068
  - iter\_type, 3065
  - put, 3067, 3068
  - time\_put, 3066
- std::time\_put\_byname, 3069
  - char\_type, 3070
  - do\_put, 3071
  - id, 3073
  - iter\_type, 3070
  - put, 3071, 3072
- std::timed\_mutex, 3073
- std::tr1, 754
- std::tr1::\_\_detail, 762
- std::tr1::\_\_is\_member\_pointer\_helper, 3074
- std::tr1::add\_const, 3075
- std::tr1::add\_cv, 3075
- std::tr1::add\_pointer, 3076
- std::tr1::add\_volatile, 3076
- std::tr1::alignment\_of, 3077
- std::tr1::array, 3078
- std::tr1::bad\_weak\_ptr, 3079
  - what, 3080
- std::tr1::extent, 3081
- std::tr1::has\_virtual\_destructor, 3082
- std::tr1::integral\_constant, 3083
- std::tr1::is\_abstract, 3085

- std::tr1::is\_arithmetic, 3086
- std::tr1::is\_array, 3087
- std::tr1::is\_class, 3088
- std::tr1::is\_compound, 3089
- std::tr1::is\_const, 3090
- std::tr1::is\_empty, 3091
- std::tr1::is\_enum, 3093
- std::tr1::is\_floating\_point, 3094
- std::tr1::is\_function, 3095
- std::tr1::is\_fundamental, 3096
- std::tr1::is\_integral, 3096
- std::tr1::is\_member\_function\_pointer, 3097
- std::tr1::is\_member\_object\_pointer, 3098
- std::tr1::is\_object, 3099
- std::tr1::is\_pointer, 3100
- std::tr1::is\_polymorphic, 3100
- std::tr1::is\_same, 3102
- std::tr1::is\_scalar, 3103
- std::tr1::is\_union, 3103
- std::tr1::is\_void, 3105
- std::tr1::is\_volatile, 3105
- std::tr1::rank, 3106
- std::tr1::remove\_all\_extents, 3108
- std::tr1::remove\_const, 3108
- std::tr1::remove\_cv, 3109
- std::tr1::remove\_extent, 3109
- std::tr1::remove\_pointer, 3110
- std::tr1::remove\_volatile, 3110
- std::try\_to\_lock\_t, 3111
- std::tuple, 3111
- std::tuple< \_T1, \_T2 >, 3112
- std::tuple\_element< 0, tuple< \_Head, \_Tail...> >, 3113
- std::tuple\_element< \_\_i, tuple< \_Head, \_Tail...> >, 3114
- std::tuple\_size< tuple< \_Elements...> >, 3114
- std::type\_info, 3115
  - ~type\_info, 3116
  - name, 3116
- std::unary\_function, 3118
  - argument\_type, 3119
  - result\_type, 3119
- std::unary\_negate, 3119
  - argument\_type, 3121
  - result\_type, 3121
- std::underflow\_error, 3122
  - what, 3122
- std::uniform\_int\_distribution, 3123
  - max, 3124
  - min, 3124
  - operator(), 3125
  - param, 3125
  - reset, 3125
  - result\_type, 3124
  - uniform\_int\_distribution, 3124
- std::uniform\_int\_distribution::param\_type, 3126
- std::uniform\_real\_distribution, 3127
  - max, 3128
  - min, 3128
  - operator(), 3128
  - param, 3129
  - reset, 3129
  - result\_type, 3128
  - uniform\_real\_distribution, 3128
- std::uniform\_real\_distribution::param\_type, 3130
- std::unique\_lock, 3130
- std::unique\_ptr, 3132
- std::unordered\_map, 3134
- std::unordered\_multimap, 3137
- std::unordered\_multiset, 3140
- std::unordered\_set, 3143
- std::valarray, 3145
  - valarray, 3149
- std::vector, 3149
  - ~vector, 3156
  - \_M\_allocate\_and\_copy, 3156
  - \_M\_range\_check, 3156
  - assign, 3157
  - at, 3158
  - back, 3159
  - begin, 3159
  - capacity, 3160
  - cbegin, 3160
  - cend, 3160
  - clear, 3160
  - crbegin, 3161
  - crend, 3161
  - data, 3161

- emplace, 3161
- empty, 3162
- end, 3162
- erase, 3162, 3163
- front, 3163, 3164
- insert, 3164–3166
- max\_size, 3166
- operator=, 3166, 3167
- pop\_back, 3168
- push\_back, 3169
- rbegin, 3169
- rend, 3169
- reserve, 3170
- resize, 3170, 3171
- shrink\_to\_fit, 3171
- size, 3171
- swap, 3171
- vector, 3153–3155
- std::vector< bool, \_Alloc >, 3172
- std::weak\_ptr, 3176
- std::weibull\_distribution, 3177
  - a, 3178
  - b, 3178
  - max, 3178
  - min, 3178
  - operator(), 3179
  - param, 3179
  - reset, 3179
  - result\_type, 3178
- std::weibull\_distribution::param\_type, 3180
- stdatomic.h, 3505
- stddev
  - std::normal\_distribution, 2799
- stdexcept, 3505
- stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf, 946, 947
- stdio\_filebuf.h, 3505
- stdio\_sync\_filebuf.h, 3506
- stl\_algo.h, 3507
- stl\_algobase.h, 3520
- stl\_bvector.h, 3523
- stl\_construct.h, 3524
- stl\_deque.h, 3524
  - \_GLIBCXX\_DEQUE\_BUF\_SIZE, 3528
- stl\_function.h, 3528
- stl\_heap.h, 3531
- stl\_iterator.h, 3533
- stl\_iterator\_base\_funcs.h, 3537
- stl\_iterator\_base\_types.h, 3538
- stl\_list.h, 3539
- stl\_map.h, 3541
- stl\_multimap.h, 3542
- stl\_multiset.h, 3543
- stl\_numeric.h, 3544
- stl\_pair.h, 3545
- stl\_queue.h, 3546
- stl\_raw\_storage\_iter.h, 3547
- stl\_relops.h, 3547
- stl\_set.h, 3548
- stl\_stack.h, 3549
- stl\_tempbuf.h, 3550
- stl\_tree.h, 3551
- stl\_uninitialized.h, 3553
- stl\_vector.h, 3555
- stossc
  - \_\_gnu\_cxx::enc\_filebuf, 892
  - \_\_gnu\_cxx::stdio\_filebuf, 964
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 992
- std::basic\_filebuf, 1530
- std::basic\_streambuf, 2022
- std::basic\_stringbuf, 2105
- str
  - std::basic\_istream, 1849
  - std::basic\_ostringstream, 1981
  - std::basic\_stringbuf, 2105, 2106
  - std::basic\_stringstream, 2166
  - std::match\_results, 2690
  - std::sub\_match, 3035
- stream\_iterator.h, 3556
- streambuf, 3557
  - io, 62
- streambuf.tcc, 3557
- streambuf\_iterator.h, 3558
- streambuf\_type
  - std::istreambuf\_iterator, 2587
  - std::ostreambuf\_iterator, 2891
- streamoff
  - std, 599
- streampos
  - std, 599

- streamsize
  - std, 599
- stride
  - numeric\_arrays, 107
- string, 3559
  - strings, 274
- string\_type
  - std::collate, 2258
  - std::collate\_byname, 2265
  - std::messages, 2701
  - std::messages\_byname, 2706
  - std::money\_get, 2714
  - std::money\_put, 2720
  - std::moneypunct, 2726
  - std::moneypunct\_byname, 2739
  - std::numpunct, 2872
  - std::numpunct\_byname, 2880
- stringbuf
  - io, 62
- stringfwd.h, 3562
- Strings, 273
- strings
  - string, 274
  - u16string, 274
  - u32string, 274
  - wstring, 274
- stringstream
  - io, 62
- substr
  - \_\_gnu\_cxx::\_\_versa\_string, 847
  - \_\_gnu\_debug::basic\_string, 1089
  - std::basic\_string, 2085
- subtractive\_rng
  - \_\_gnu\_cxx::subtractive\_rng, 999
- suffix
  - std::match\_results, 2690
- sum
  - numeric\_arrays, 107
- sungetc
  - \_\_gnu\_cxx::enc\_filebuf, 892
  - \_\_gnu\_cxx::stdio\_filebuf, 965
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 992
  - std::basic\_filebuf, 1530
  - std::basic\_streambuf, 2022
  - std::basic\_stringbuf, 2106
- swap
  - \_\_gnu\_cxx, 372
  - \_\_gnu\_cxx::\_\_versa\_string, 848
  - \_\_gnu\_debug::basic\_string, 1090
  - mutating\_algorithms, 145
  - regex, 249, 250
  - std, 679–681
  - std::basic\_regex, 2002
  - std::basic\_string, 2086
  - std::deque, 2398
  - std::forward\_list, 2455
  - std::function<
    - \_Res(\_-
      - ArgTypes...)>, 2469
  - std::list, 2629
  - std::map, 2675
  - std::match\_results, 2690
  - std::multimap, 2766
  - std::multiset, 2785
  - std::set, 2995
  - std::vector, 3171
- swap\_ranges
  - mutating\_algorithms, 145
- sync
  - \_\_gnu\_cxx::enc\_filebuf, 892
  - \_\_gnu\_cxx::stdio\_filebuf, 965
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 992
  - std::basic\_filebuf, 1531
  - std::basic\_fstream, 1593
  - std::basic\_ifstream, 1648
  - std::basic\_iostream, 1742
  - std::basic\_istream, 1794
  - std::basic\_istreamstream, 1849
  - std::basic\_streambuf, 2023
  - std::basic\_stringbuf, 2107
  - std::basic\_stringstream, 2167
- sync\_with\_stdio
  - std::basic\_fstream, 1594
  - std::basic\_ifstream, 1648
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1743
  - std::basic\_istream, 1795
  - std::basic\_istreamstream, 1850
  - std::basic\_ofstream, 1894
  - std::basic\_ostream, 1936
  - std::basic\_ostreamstream, 1981
  - std::basic\_stringstream, 2167
  - std::ios\_base, 2561

- syntax\_option\_type
  - std::regex\_constants, 743
- system\_error, 3563
- t
  - std::binomial\_distribution, 2193
- table
  - std::ctype< char >, 2307
  - std::ctype\_byname< char >, 2357
- table\_size
  - std::ctype< char >, 2310
  - std::ctype\_byname< char >, 2360
- tag\_and\_trait.hpp, 3564
- tags.h, 3566
- tan
  - complex\_numbers, 53
- tanh
  - complex\_numbers, 53
- target
  - std::function< \_Res(\_- ArgTypes...)>, 2469
- target\_type
  - std::function< \_Res(\_- ArgTypes...)>, 2470
- tellg
  - std::basic\_fstream, 1594
  - std::basic\_ifstream, 1649
  - std::basic\_iostream, 1743
  - std::basic\_istream, 1795
  - std::basic\_istreamstream, 1850
  - std::basic\_stringstream, 2167
- tellp
  - std::basic\_fstream, 1595
  - std::basic\_iostream, 1743
  - std::basic\_ofstream, 1894
  - std::basic\_ostream, 1937
  - std::basic\_ostringstream, 1982
  - std::basic\_stringstream, 2168
- temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer, 1001
- terminate
  - exceptions, 37
- terminate\_handler
  - exceptions, 35
- test
  - std::bitset, 2207
- tgmath.h, 3568
- thousands\_sep
  - std::moneypunct, 2735
  - std::moneypunct\_byname, 2747
  - std::numpunct, 2876
  - std::numpunct\_byname, 2883
- thread, 3568
- Threads, 75
- throw\_allocator.h, 3570
- throw\_with\_nested
  - exceptions, 37
- tie
  - std::basic\_fstream, 1595
  - std::basic\_ifstream, 1649
  - std::basic\_ios, 1681
  - std::basic\_iostream, 1744
  - std::basic\_istream, 1796
  - std::basic\_istreamstream, 1851
  - std::basic\_ofstream, 1895
  - std::basic\_ostream, 1937
  - std::basic\_ostringstream, 1982, 1983
  - std::basic\_stringstream, 2168, 2169
- Time, 38
- time
  - std::locale, 2639
- time\_get
  - std::time\_get, 3044
- time\_members.h, 3572
- time\_point\_cast
  - std::chrono, 730
- time\_put
  - std::time\_put, 3066
- tinyness\_before
  - std::\_\_numeric\_limits\_base, 1398
  - std::numeric\_limits, 2845
- TLB\_size
  - \_\_gnu\_parallel::\_Settings, 1213
- to\_string
  - std::bitset, 2207
- to\_ulong
  - std::bitset, 2208
- tolower
  - std, 682
  - std::\_\_ctype\_abstract\_base, 1321
  - std::ctype, 2293, 2294
  - std::ctype< char >, 2307, 2308

- std::ctype< wchar\_t >, 2329
- std::ctype\_byname, 2344, 2345
- std::ctype\_byname< char >, 2357, 2358
- top
  - std::priority\_queue, 2926
  - std::stack, 3026
- toupper
  - std, 682
  - std::\_\_ctype\_abstract\_base, 1321, 1322
  - std::ctype, 2294, 2295
  - std::ctype< char >, 2308, 2309
  - std::ctype< wchar\_t >, 2329, 2330
  - std::ctype\_byname, 2345, 2346
  - std::ctype\_byname< char >, 2358, 2359
- trl\_math\_spec\_func
  - assoc\_laguerre, 111
  - assoc\_legendre, 111
  - beta, 111
  - comp\_ellint\_1, 111
  - comp\_ellint\_2, 112
  - comp\_ellint\_3, 112
  - conf\_hyperg, 112
  - cyl\_bessel\_i, 112
  - cyl\_bessel\_j, 112
  - cyl\_bessel\_k, 112
  - cyl\_neumann, 113
  - ellint\_1, 113
  - ellint\_2, 113
  - ellint\_3, 113
  - expint, 113
  - hermite, 114
  - hyperg, 114
  - laguerre, 114
  - legendre, 114
  - riemann\_zeta, 114
  - sph\_bessel, 114
  - sph\_legendre, 114
  - sph\_neumann, 115
- traits\_type
  - \_\_gnu\_cxx::enc\_filebuf, 878
  - \_\_gnu\_cxx::stdio\_filebuf, 946
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 978
  - std::basic\_filebuf, 1513
  - std::basic\_fstream, 1552, 1553
  - std::basic\_ifstream, 1616
  - std::basic\_ios, 1667
  - std::basic\_iostream, 1703
  - std::basic\_istream, 1764
  - std::basic\_istream::sentry, 1806
  - std::basic\_istreamstream, 1818
  - std::basic\_ofstream, 1871
  - std::basic\_ostream, 1914
  - std::basic\_ostreamstream, 1959
  - std::basic\_streambuf, 2008
  - std::basic\_stringbuf, 2091
  - std::basic\_stringstream, 2126, 2127
  - std::istreambuf\_iterator, 2587
  - std::ostream\_iterator, 2887
  - std::ostreambuf\_iterator, 2892
- transform
  - mutating\_algorithms, 145, 146
  - std::collate, 2261
  - std::collate\_byname, 2267
  - std::regex\_traits, 2964
- transform\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1213
- transform\_primary
  - std::regex\_traits, 2965
- translate
  - std::regex\_traits, 2965
- translate\_nocase
  - std::regex\_traits, 2965
- traps
  - std::\_\_numeric\_limits\_base, 1399
  - std::numeric\_limits, 2845
- tree\_policy.hpp, 3573
- tree\_trace\_base.hpp, 3573
- trie\_policy.hpp, 3573
- true\_type
  - metaprogramming, 121
- truname
  - std::numpunct, 2877
  - std::numpunct\_byname, 2883
- trunc
  - std::basic\_fstream, 1605
  - std::basic\_ifstream, 1658
  - std::basic\_ios, 1689
  - std::basic\_iostream, 1753
  - std::basic\_istream, 1804

- std::basic\_istream, 1860
- std::basic\_ofstream, 1904
- std::basic\_ostream, 1946
- std::basic\_ostringstream, 1991
- std::basic\_stringstream, 2178
- std::ios\_base, 2569
- try\_lock
  - mutexes, 71
- tuple, 3574
- Type Traits, 115
- type\_traits, 3576, 3579
- type\_traits.h, 3582
- type\_utils.hpp, 3583
- typeinfo, 3583
- typelist.h, 3584
- types.h, 3585
- types\_traits.hpp, 3587
- u16streampos
  - std, 599
- u16string
  - strings, 274
- u32streampos
  - std, 599
- u32string
  - strings, 274
- uflow
  - \_\_gnu\_cxx::enc\_filebuf, 893
  - \_\_gnu\_cxx::stdio\_filebuf, 965
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 993
  - std::basic\_filebuf, 1531
  - std::basic\_streambuf, 2023
  - std::basic\_stringbuf, 2107
- uncaught\_exception
  - exceptions, 37
- underflow
  - \_\_gnu\_cxx::enc\_filebuf, 893
  - \_\_gnu\_cxx::stdio\_filebuf, 966
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 993
  - std::basic\_filebuf, 1531
  - std::basic\_streambuf, 2023
  - std::basic\_stringbuf, 2107
- unexpected
  - exceptions, 38
- unexpected\_handler
  - exceptions, 35
- unget
  - std::basic\_fstream, 1596
  - std::basic\_ifstream, 1650
  - std::basic\_iostream, 1745
  - std::basic\_istream, 1796
  - std::basic\_istream, 1851
  - std::basic\_stringstream, 2169
- Uniform, 284
- uniform\_int\_distribution
  - std::uniform\_int\_distribution, 3124
- uniform\_real\_distribution
  - std::uniform\_real\_distribution, 3128
- uninitialized\_copy
  - std, 682
- uninitialized\_copy\_n
  - SGIextensions, 25
  - std, 682
- uninitialized\_fill
  - std, 683
- uninitialized\_fill\_n
  - std, 683
- unique
  - mutating\_algorithms, 147
  - std::forward\_list, 2455
  - std::list, 2629
- unique\_copy
  - mutating\_algorithms, 148
- unique\_copy.h, 3587
- unique\_copy\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1213
- unique\_ptr.h, 3588
- unitbuf
  - std, 684
  - std::basic\_fstream, 1605
  - std::basic\_ifstream, 1658
  - std::basic\_ios, 1689
  - std::basic\_iostream, 1754
  - std::basic\_istream, 1805
  - std::basic\_istream, 1860
  - std::basic\_ofstream, 1904
  - std::basic\_ostream, 1946
  - std::basic\_ostringstream, 1991
  - std::basic\_stringstream, 2178
  - std::ios\_base, 2569
- Unordered Associative, 30
- unordered\_map, 3589, 3590



- unordered\_map.h, 3592
- unordered\_set, 3594, 3595
- unordered\_set.h, 3596
- unsetf
  - std::basic\_fstream, 1596
  - std::basic\_ifstream, 1650
  - std::basic\_ios, 1681
  - std::basic\_istream, 1745
  - std::basic\_istream, 1797
  - std::basic\_istream, 1852
  - std::basic\_ofstream, 1896
  - std::basic\_ostream, 1938
  - std::basic\_ostringstream, 1983
  - std::basic\_stringstream, 2170
  - std::ios\_base, 2562
- unshift
  - std::\_\_codecvt\_abstract\_base, 1307
  - std::codecvt, 2231
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2237
  - std::codecvt< char, char, mbstate\_t >, 2242
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2247
  - std::codecvt\_byname, 2254
- upper\_bound
  - binary\_search\_algorithms, 198, 199
  - std::map, 2676
  - std::multimap, 2767
  - std::multiset, 2785
  - std::set, 2995, 2996
- uppercase
  - std, 684
  - std::basic\_fstream, 1605
  - std::basic\_ifstream, 1658
  - std::basic\_ios, 1689
  - std::basic\_istream, 1754
  - std::basic\_istream, 1805
  - std::basic\_istream, 1860
  - std::basic\_ofstream, 1904
  - std::basic\_ostream, 1946
  - std::basic\_ostringstream, 1991
  - std::basic\_stringstream, 2178
  - std::ios\_base, 2569
- use\_facet
  - std, 684
- std::locale, 2637
- std::locale::id, 2643
- Utilities, 76
- utility, 3598
- valarray, 3599
  - numeric\_arrays, 88–90
  - std::valarray, 3149
- valarray\_after.h, 3604
- valarray\_array.h, 3619
- valarray\_array.tcc, 3630
- valarray\_before.h, 3631
- value
  - regex, 250
- value\_comp
  - std::map, 2676
  - std::multimap, 2767
  - std::multiset, 2786
  - std::set, 2996
- value\_compare
  - std::set, 2982
- value\_type
  - std::back\_insert\_iterator, 1501
  - std::complex, 2270
  - std::front\_insert\_iterator, 2461
  - std::insert\_iterator, 2548
  - std::istream\_iterator, 2583
  - std::istreambuf\_iterator, 2587
  - std::iterator, 2591
  - std::ostream\_iterator, 2887
  - std::ostreambuf\_iterator, 2892
  - std::raw\_storage\_iterator, 2944
  - std::reverse\_iterator, 2969
  - std::set, 2982
- vector, 3631, 3632
  - std::\_\_debug::vector, 1383
  - std::vector, 3153–3155
- vector.tcc, 3634
- vstring.h, 3634
- vstring.tcc, 3638
- vstring\_fwd.h, 3639
- vstring\_util.h, 3640
- wcerr
  - std, 686
- wcin

- std, 686
- wclog
  - std, 686
- wcout
  - std, 686
- wcregex\_token\_iterator
  - regex, 221
- wcsub\_match
  - regex, 221
- wfilebuf
  - io, 62
- wfstream
  - io, 62
- what
  - \_\_gnu\_cxx::forced\_error, 903
  - \_\_gnu\_cxx::recursive\_init\_error, 928
  - std::bad\_alloc, 1503
  - std::bad\_cast, 1504
  - std::bad\_exception, 1505
  - std::bad\_function\_call, 1506
  - std::bad\_typeid, 1508
  - std::domain\_error, 2412
  - std::exception, 2420
  - std::future\_error, 2479
  - std::invalid\_argument, 2550
  - std::ios\_base::failure, 2570
  - std::length\_error, 2595
  - std::logic\_error, 2646
  - std::out\_of\_range, 2894
  - std::overflow\_error, 2896
  - std::range\_error, 2936
  - std::regex\_error, 2949
  - std::runtime\_error, 2975
  - std::system\_error, 3037
  - std::tr1::bad\_weak\_ptr, 3080
  - std::underflow\_error, 3122
- widen
  - std::\_\_ctype\_abstract\_base, 1322, 1323
  - std::basic\_fstream, 1597
  - std::basic\_ifstream, 1651
  - std::basic\_ios, 1682
  - std::basic\_iostream, 1745
  - std::basic\_istream, 1797
  - std::basic\_istream, 1852
  - std::basic\_ofstream, 1896
  - std::basic\_ostream, 1938
  - std::basic\_ostringstream, 1983
  - std::basic\_stringstream, 2170
  - std::ctype, 2295, 2296
  - std::ctype< char >, 2309, 2310
  - std::ctype< wchar\_t >, 2330, 2331
  - std::ctype\_byname, 2346, 2347
  - std::ctype\_byname< char >, 2359, 2360
- width
  - std::basic\_fstream, 1597
  - std::basic\_ifstream, 1651
  - std::basic\_ios, 1682
  - std::basic\_iostream, 1746
  - std::basic\_istream, 1798
  - std::basic\_istream, 1853
  - std::basic\_ofstream, 1896, 1897
  - std::basic\_ostream, 1939
  - std::basic\_ostringstream, 1984
  - std::basic\_stringstream, 2170, 2171
  - std::ios\_base, 2562
- wfstream
  - io, 63
- wios
  - io, 63
- wiostream
  - io, 63
- wistream
  - io, 63
- wistringstream
  - io, 63
- wofstream
  - io, 63
- workstealing.h, 3640
- wostream
  - io, 63
- wostringstream
  - io, 64
- wregex
  - regex, 221
- write
  - std::basic\_fstream, 1598
  - std::basic\_iostream, 1747
  - std::basic\_ofstream, 1897
  - std::basic\_ostream, 1939

- [std::basic\\_ostringstream](#), [1984](#)
    - [std::basic\\_stringstream](#), [2171](#)
  - [ws](#)
    - [std](#), [685](#)
  - [wsregex\\_token\\_iterator](#)
    - [regex](#), [221](#)
  - [wssub\\_match](#)
    - [regex](#), [221](#)
  - [wstreambuf](#)
    - [io](#), [64](#)
  - [wstreampos](#)
    - [std](#), [599](#)
  - [wstring](#)
    - [strings](#), [274](#)
  - [wstringbuf](#)
    - [io](#), [64](#)
  - [wstringstream](#)
    - [io](#), [64](#)
- [xalloc](#)
  - [std::basic\\_fstream](#), [1598](#)
  - [std::basic\\_ifstream](#), [1652](#)
  - [std::basic\\_ios](#), [1683](#)
  - [std::basic\\_iostream](#), [1747](#)
  - [std::basic\\_istream](#), [1798](#)
  - [std::basic\\_istreamstream](#), [1853](#)
  - [std::basic\\_ofstream](#), [1898](#)
  - [std::basic\\_ostream](#), [1940](#)
  - [std::basic\\_ostringstream](#), [1985](#)
  - [std::basic\\_stringstream](#), [2172](#)
  - [std::ios\\_base](#), [2563](#)
- [xsgetn](#)
  - [\\_\\_gnu\\_cxx::enc\\_filebuf](#), [894](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf](#), [967](#)
  - [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf](#), [994](#)
  - [std::basic\\_filebuf](#), [1532](#)
  - [std::basic\\_streambuf](#), [2024](#)
  - [std::basic\\_stringbuf](#), [2108](#)
- [xspn](#)
  - [\\_\\_gnu\\_cxx::enc\\_filebuf](#), [894](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf](#), [967](#)
  - [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf](#), [994](#)
  - [std::basic\\_filebuf](#), [1533](#)
  - [std::basic\\_streambuf](#), [2025](#)
  - [std::basic\\_stringbuf](#), [2109](#)
- [yield](#)
  - [std::this\\_thread](#), [754](#)