

# Prototype reimplementaion of $\text{\LaTeX} 2_{\epsilon}$ 's block environments using templates

$\text{\LaTeX}$  Project\*

v0.9j 2025-07-09

## Abstract

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Template types and templates for blocks and lists</b>	<b>3</b>
2.1	Template types . . . . .	3
2.1.1	The template type ‘block’ . . . . .	3
2.1.2	The template type ‘para’ . . . . .	4
2.1.3	The template type ‘list’ . . . . .	4
2.1.4	The template type ‘item’ . . . . .	4
2.1.5	The template type ‘blockenv’ . . . . .	4
2.2	Templates . . . . .	5
2.2.1	The <code>blockenv</code> template ‘display’ . . . . .	5
2.2.2	The <code>block</code> template ‘display’ . . . . .	6
2.2.3	The <code>para</code> template ‘std’ . . . . .	7
2.2.4	The <code>list</code> template ‘std’ . . . . .	7
2.2.5	The <code>item</code> template ‘std’ . . . . .	8
<b>3</b>	<b>Declaration of standard block environments</b>	<b>9</b>
3.1	The <code>center</code> , <code>flushleft</code> , and <code>flushright</code> environments . . . . .	9
3.2	The <code>quote</code> and <code>quotation</code> environments . . . . .	9
3.3	The <code>verbatim</code> and <code>verbatim*</code> environments . . . . .	9
3.4	The <code>itemize</code> environment . . . . .	10
3.5	The <code>enumerate</code> environment . . . . .	10
3.6	The <code>description</code> environment . . . . .	10
3.7	The <code>list</code> environment . . . . .	11
3.8	The <code>verse</code> environment . . . . .	11
3.9	The <code>trivlist</code> environment . . . . .	11
3.10	Environments declared through <code>\newtheorem</code> . . . . .	11
<b>4</b>	<b>Adjusting the layout of standard block environments</b>	<b>11</b>

---

\*Initial reimplementaion of lists done by Bruno Le Floch, generalized second version with tagging support by Frank Mittelbach.

<b>5</b>	<b>Tagging support</b>	<b>12</b>
5.1	Paragraph tags . . . . .	12
5.2	Tagging recipes . . . . .	13
<b>6</b>	<b>Debugging</b>	<b>15</b>
<b>7</b>	<b>New and redefined kernel command</b>	<b>15</b>
<b>8</b>	<b>The Implementation</b>	<b>16</b>
8.1	Handling <code>\par</code> after the end of the list . . . . .	16
8.2	Template types and template interfaces . . . . .	18
8.3	Useful helper commands . . . . .	20
8.3.1	Debugging . . . . .	21
8.4	Implementation of templates . . . . .	22
8.4.1	Implementation of <code>blockenv</code> templates ... . . . . .	22
8.4.2	Implementation of <code>para</code> templates ... . . . . .	28
8.4.3	Implementation of <code>block</code> templates ... . . . . .	28
8.4.4	Implementation of <code>list</code> templates ... . . . . .	31
8.4.5	Implementation of <code>\item</code> template(s) . . . . .	35
8.5	Tagging support commands . . . . .	43
8.5.1	List tags . . . . .	47
8.6	Tagging recipes . . . . .	49
<b>9</b>	<b>Implementation of document-level block environments</b>	<b>52</b>
9.1	<code>Displayblock</code> environments . . . . .	52
9.2	The <code>center</code> , <code>flushleft</code> , and <code>flushright</code> environments . . . . .	52
9.3	<code>Display quote</code> environments . . . . .	53
9.4	<code>Verbatim</code> environments . . . . .	53
9.4.1	Helper commands for <code>verbatim</code> . . . . .	53
9.5	<code>Standard list</code> environments . . . . .	54
9.6	<code>verse</code> environment . . . . .	54
9.7	<code>Theorem-like</code> environments . . . . .	57
<b>10</b>	<b>Instance declarations for environments</b>	<b>59</b>
10.1	<code>Blockenv</code> instances . . . . .	59
10.1.1	Basic instances . . . . .	60
10.1.2	<code>Center</code> , <code>flushleft</code> , and <code>flushright</code> instances . . . . .	60
10.1.3	<code>Blockquote</code> instances . . . . .	61
10.1.4	The <code>theorem</code> instance . . . . .	62
10.1.5	The <code>verbatim</code> and <code>verbatim*</code> instances . . . . .	63
10.1.6	<code>Standard list</code> instances . . . . .	64
10.2	<code>Block</code> instances . . . . .	65
10.2.1	<code>Displayblock</code> instances . . . . .	65
10.2.2	<code>Verbatim</code> instances . . . . .	66
10.2.3	<code>Quote/quotationblock</code> instances . . . . .	66
10.2.4	<code>Block</code> instances for the theorems . . . . .	67
10.2.5	<code>Block</code> instances for the standard lists . . . . .	67
10.3	<code>List</code> instances for the standard lists . . . . .	67
10.4	<code>Item</code> instances . . . . .	68
10.5	<code>Para</code> instances . . . . .	69

<b>A</b>	<b>Documentation from first prototype implementations</b>	<b>70</b>
	A.1 Open questions . . . . .	70
	A.2 Code cleanup . . . . .	70
	A.3 Tasks . . . . .	70
<b>B</b>	<b>Plan of attack of first prototype</b>	<b>71</b>
	<b>Index</b>	<b>73</b>

## 1 Introduction

The list implementation in  $\text{\LaTeX} 2_{\epsilon}$  serves a dual purpose: it implements real lists such as `itemize` or `enumerate`, but it is also used as the basis for vertical blocks, i.e., to specify the vertical spacing and paragraph handling after such block, e.g., in environments like `center`, `quote`, `verbatim`, or in the theorem environments. They are all implemented as “trivial” lists with a single (hidden) item.

While this was convenient to get a consistent layout using a single implementation it is not adequate if it comes to interpreting the structure of a document, because environments based on `trivlist` should not advertise themselves as being a “list” — after all, from a semantic point of view they aren’t lists.

The approach taking here is therefore to offer separate template types: `block` (horizontally or vertically oriented data that needs some handling at the start and the end), `para` (that deals with different paragraph layouts), `list` (that handles list related parameters, and `item` (for item layouts and handling). To address the independent aspects we have the template type `blockenv` that ties them together as necessary.

For example, a `quote` environment would make use of a (display) `block` and some `para` instance while an standard `enumerate` would make use of a display `block`, a `list`, and an `item` and `para` instance. An inline list (like `enumerate*` from the `enumitem` package) would be using the same `list` instance but a different (horizontally oriented) `block` instance build from a different template.

## 2 Template types and templates for blocks and lists

### 2.1 Template types

#### 2.1.1 The template type ‘block’

**Arg:** 1 key/value list to alter the default block parameters

#### Semantics:

Handle the layout aspects of a block of data. In case of a “display” block (i.e., vertically oriented) the spacing and page breaking as well as the handling if the block starts a paragraph or ends one, that is, if text is immediately following the block without being separated by an empty line, then this text is considered to be in the same paragraph as the block.

In case of a horizontally oriented block it covers any special handling at the start and end of the block, e.g, extra spacing, prohibiting or encouraging line breaks, and so forth.

### 2.1.2 The template type ‘para’

**Arg: 1** key/value list to alter the default item parameters

#### **Semantics:**

Sets up paragraph-specific parameters for H&J, e.g., to implement justification variations, the behavior of \\ etc. The instances are used in higher-level templates, e.g., in a `block`.

### 2.1.3 The template type ‘list’

**Arg: 1** key/value list to alter the default item parameters

#### **Semantics:**

Handle the aspects related to list design, e.g., the use and formatting of counters, etc.

Note that this does not cover block-related aspects, i.e., a list instance could be used both for a display list or for an inline line.

### 2.1.4 The template type ‘item’

**Arg: 1** key/value list to alter the default item parameters

#### **Semantics:**

A sub-type used as part of `list` to easily cover alternative layout for list items.

### 2.1.5 The template type ‘blockenv’

**Arg: 1** key/value list to alter the default parameters of the template instances used by the particular block environment

#### **Semantics:**

This template type is used to implement document-level environments. It defines a `block` instance to handle the layout at the “edge” of the environment data, possibly some paragraph setup through a `para` instance, potentially an “inner” instance for more complicated environments (such as lists), and possibly some additional setup code for certain environments.

It also defines how the `blockenv` behaves with respect to nesting, e.g., does it change when nested and if so how many levels of nesting are supported, etc.

Finally, the template type defines how it appears in a tagged PDF document, what tag names are used, how they are rolemapped and whether it adds additional attributes, etc.

## 2.2 Templates

### 2.2.1 The `blockenv` template ‘display’

#### Attributes:

- name** (*tokenlist*) Name of the environment used in tracing and error messages.
- tag-name** (*tokenlist*) Name of the tag used for the block inside the PDF. If not explicitly given the name is defined by the `tagging-recipe`. Note that in case of `tagging-recipe=basic` no tag for the block is produced, so any key settings are ignored.  
Default: `<empty>`
- tag-attr-class** (*tokenlist*) An explicit tag class attribute. Default: `<empty>`
- tagging-recipe** (*tokenlist*) Defines the way tagging is done. Currently the values `basic`, `standard`, and `list` are supported. Default: `standard`
- increment-level** (*boolean*) Does this `blockenv` increase the block level if it is nested in an outer block? Default: `true`
- setup-code** (*tokenlist*) Initial setup code. This is executed after legacy defaults (from `\@listi`, `\@listii`, etc.) are used but before the block instance is called.  
Default: `<empty>`
- block-instance** (*tokenlist*) Part of the name of the `block` instance that is called. The full name has a `-<level>` appended. Default: `displayblock`
- para-instance** (*tokenlist*) Paragraph settings to use within the environment. If `<empty>` then outer values are retained. However, the block template resets some values, which may not be the right thing to do. Default: `<empty>`
- inner-level-counter** (*tokenlist*) Name of an existing (!) counter that is incremented and used to determine final name of the `inner-instance` or empty if always the same inner instance should be used.
- max-inner-levels** (*tokenlist*) Maximum number of nested environments of this kind. Only relevant if there is a `inner-level-counter` specified. Default: 4
- inner-instance-type** (*tokenlist*) Template type of the inner instance. Default: `list`
- inner-instance** (*tokenlist*) Name of the inner instance (if any). If there is an `inner-level-counter` then the instance name gets `-<counter value>` appended.  
Default: `<empty>`
- tagging-suppress-paras** (*boolean*) *describe* Default: `false`
- final-code** (*tokenlist*) Final setup code Default: `\ignorespaces`

**Semantics & Comments:** This `blockenv` template supports the legacy list setting that are found in many document classes in the macros `\@listi`, `\@listii`, up to `\@listvi`. It also uses the counter `\@listdepth` to track nesting of block, again mainly to support legacy setups (internally it gives it a more appropriate name but it remains accessible through the  $\LaTeX 2\epsilon$  name).

It first checks that nothing is too deeply nested. If the level should increase then it increments the `\@listdepth` counter and calls the corresponding `\@list...` macro to update the legacy defaults. If `increment-level` is set to `false` this is bypassed.

It then sets up the tagging via the `tagging-recipe` setting and executes any code in `setup-code`.

Afterwards it calls the appropriate `block` instance based on `block-instance` and current level, e.g., `displayblock-1`. Then it sets up paragraph parameters if a `para-instance` was specified (otherwise they stay as they are).

If a `inner-instance` was specified this is called next, or more precisely: if no `inner-level-counter` was specified the instance `inner-instance` is called.

Otherwise, the `inner-level-counter` is incremented and the instance with the name `inner-instance-inner-level-counter` is called.

Finally, the `final-code` is executed (by default `\ignorespaces`).

The maximum number of `blockenvs` that can be nested into each other is restricted by the  $\LaTeX$  counter `maxblocklevels` with a default value of 6. If this value is increased then it is necessary to provide additional instances, e.g., `displayblock-7`, etc. Decreasing is, of course, always possible, then some of the instances defined are not used and instead the user gets an error that there is too much nesting going on.

If the key `increment-level` is set to `false` then such an environment doesn't alter the nesting level and therefore you can nest those environments as often as you like (a typical example would be `flushleft` anywhere in the nesting hierarchy, that would have no effect on hitting the boundary).

### 2.2.2 The block template ‘display’

**Attributes:**

<code>begin-vspace</code> ( <i>skip</i> )	Default: <code>\topsep</code>
<code>begin-extra-vspace</code> ( <i>skip</i> )	Default: <code>\partopsep</code>
<code>para-vspace</code> ( <i>skip</i> )	Default: <code>\parsep</code>
<code>end-vspace</code> ( <i>skip</i> )	Default: value from <code>begin-vspace</code>
<code>end-extra-vspace</code> ( <i>skip</i> )	Default: value from <code>begin-extra-vspace</code>
<code>item-vspace</code> ( <i>skip</i> )	The space in front of an item if the block is a list; if not the setting has no effect
	Default: <code>\itemsep</code>
<code>begin-penalty</code> ( <i>integer</i> )	Default: <code>\@beginparpenalty</code>
<code>end-penalty</code> ( <i>integer</i> )	Default: <code>\@endparpenalty</code>
<code>left-margin</code> ( <i>length</i> )	Default: <code>\leftmargin</code>
<code>right-margin</code> ( <i>length</i> )	Default: <code>\rightmargin</code>
<code>para-indent</code> ( <i>length</i> )	Default: <code>0pt</code>

**Semantics & Comments:** The idea of a `heading` key needs some further thoughts and therefore has been removed for now. Maybe instead the template type should accept a second argument and receive input for such a heading from the document level instead.

The names of the keys need further thoughts and some decision. Right now it is a mixture of those with hyphens and those that match legacy register names (the way `enumitem` did its keys).

### 2.2.3 The `para` template ‘std’

**Attributes:**

<code>para-indent</code> ( <i>length</i> )	Default: <code>\parindent</code>
<code>begin-hspace</code> ( <i>skip</i> )	Default: <code>0pt</code>
<code>left-hspace</code> ( <i>skip</i> )	Default: <code>0pt</code>
<code>right-hspace</code> ( <i>skip</i> )	Default: <code>0pt</code>
<code>end-hspace</code> ( <i>skip</i> )	Default: <code>\@flushglue</code>
<code>fixed-word-spaces</code> ( <i>boolean</i> )	Default: <code>false</code>
<code>final-hyphen-demerits</code> ( <i>integer</i> )	Default: <code>5000</code>
<code>newline-cmd</code> ( <i>tokenlist</i> )	Default: <code>\@normalcr</code>
<code>para-attr-class</code> ( <i>tokenlist</i> )	Default: <code>justify</code>

### 2.2.4 The `list` template ‘std’

**Attributes:**

<code>counter</code> ( <i>tokenlist</i> )	Counter name to be used in a numbered list or empty, if the list is unnumbered
<code>item-label</code> ( <i>tokenlist</i> )	Label “string” for a fixed label or as generated from the current <code>counter</code> value
<code>start</code> ( <i>integer</i> )	Start value for the counter if the list is numbered, otherwise irrelevant Default: <code>1</code>
<code>resume</code> ( <i>boolean</i> )	Should a numbered list be resumed from the last instance? Default: <code>false</code>
<code>item-instance</code> ( <i>instance</i> )	Instance of type <code>item</code> to be used to format the label string Default: <code>basic</code>
<code>item-vspace</code> ( <i>skip</i> )	The space in front of an item in the list. If not specified the value specified in the block template instance is used

<b>item-indent</b> ( <i>length</i> )	Horizontal displacement of the item.	Default: 0pt
<b>item-penalty</b> ( <i>integer</i> )	Penalty for breaking before an item (except the first)	Default: \@itempenalty
<b>label-width</b> ( <i>length</i> )	Width reserved for the formatted item label	Default: \labelwidth
<b>label-sep</b> ( <i>length</i> )	Horizontal separation between label and following text	Default: \labelsep
<b>legacy-support</b> ( <i>boolean</i> )	Is formatting the label via \makelabel supported?	Default: false

### 2.2.5 The item template ‘std’

#### Attributes:

<b>counter-label</b> ( <i>function1</i> )	<i>unused</i>	Default: \arabic{#1}
<b>counter-ref</b> ( <i>function1</i> )	<i>unused</i>	Default: value from counter-label
<b>label-ref</b> ( <i>function1</i> )	<i>unused</i>	Default: #1
<b>label-autoref</b> ( <i>function1</i> )	<i>unused</i>	Default: item #1
<b>label-format</b> ( <i>function1</i> )	Formatting of the label, questionable the way it is used	Default: #1
<b>label-strut</b> ( <i>boolean</i> )	Add a \strut to the label?	Default: false
<b>label-align</b> ( <i>choice</i> )	Supported values left, center, right, and parleft. <i>Only partly implemented</i>	Default: right
<b>label-boxed</b> ( <i>boolean</i> )	Should the label be boxed?	Default: true
<b>next-line</b> ( <i>boolean</i> )		Default: false
<b>text-font</b> ( <i>tokenlist</i> )	<i>unused</i>	
<b>compatibility</b> ( <i>boolean</i> )		Default: true

**Semantics & Comments:** This template is only rudimentary implemented at the moment. It probably needs other keys and the existing ones need a proper implementation.



## 3 Declaration of standard block environments

### 3.1 The center, flushleft, and flushright environments

The `center` environment is defined through the `blockenv` instance `center` which makes use of the `block` instance `displayblock-⟨level⟩` and the `para` instance `center`. The block nesting level is not incremented. With respect to tagging, text separated by `\par` commands (or empty lines) inside the environment is not tagged as separate paragraphs, i.e., the whole environment is considered to be part of an outer paragraph. The default implementation is

```
\DeclareInstance{blockenv}{center}{display}
{
  name           = center,
  tagging-recipe = basic,
  tagging-suppress-paras = true ,
  increment-level = false,
  block-instance = displayblock ,
  para-instance  = center ,
}
```

The `flushleft` and `flushright` environments are defined in a similar way.

### 3.2 The quote and quotation environments

The `quote` environment is defined through the `blockenv` instance `quote` which makes use of the `block` instance `quoteblock-⟨level⟩`. The paragraph setup is inherited. The block nesting level is incremented. The default implementation is

```
\DeclareInstance{blockenv}{quote}{display}
{
  name           = quote,
  tag-name       = quote,
  tagging-recipe = standard,
  increment-level = true,
  block-instance = quoteblock ,
}
```

The implementation of `quotation` is similar but uses `quotationblock-⟨level⟩`.

### 3.3 The verbatim and verbatim\* environments

Both the `verbatim` environment is defined through the `blockenv` instance `verbatim` which makes use of the `block` instance `verbatimblock-⟨level⟩` and the `para` instance `justify`. The block nesting level is not incremented. Verbatim processing requires various catcode changes, etc. and as a consequence a special parsing routine that grabs the whole environment while these catcodes are in force. This setup is done in the `final-code` key and its last action is to initiate the special parsing. The default implementation is

```

\DeclareInstance{blockenv}{verbatim}{display}
{
  name                = verbatim,
  tag-name            = verbatim,
  tagging-recipe      = standard,
  tagging-suppress-paras = true,
  increment-level     = false,
  block-instance      = verbatimblock ,
  para-instance       = justify ,
  final-code          = \legacyverbatimsetup
                      \@setupverbinvisiblespace \@vobeyspaces
                      \@xverbatim
}

```

The implementation of `verbatim*` is similar using the `blockenv` instance `verbatim*`. Its `final-code` sets up visible spaces and a slightly different parsing that grabs everything up to `\end{verbatim*}`. Otherwise the setup is identical.

### 3.4 The `itemize` environment

The `itemize` environment is defined through the `blockenv` instance `itemize` which makes use of the `block` instance `list-⟨level⟩`, and an inner instance `itemize-⟨inner-level⟩` of type `list`. The paragraph setup is inherited. The `⟨inner-level⟩` is determined through `\@itemdepth`. The block nesting level and the inner list nesting level are incremented. The default implementation is

```

\DeclareInstance{blockenv}{itemize}{display}
{
  name                = itemize,
  tag-name            = itemize,
  tag-attr-class      = itemize,
  tagging-recipe      = list,
  inner-level-counter = \@itemdepth,
  increment-level     = true,
  max-inner-levels    = 4,
  block-instance      = listblock ,
  inner-instance      = itemize ,
}

```

### 3.5 The `enumerate` environment

The `enumerate` environment is similar to `itemize` but uses the `blockenv` instance `enumerate`, the `block` instance `list-⟨level⟩`, and the inner instance `enumerate-⟨inner-level⟩`. The `⟨inner-level⟩` is determined through `\@enumdepth`.

### 3.6 The `description` environment

The `description` environment uses the `blockenv` instance `description`, the `block` instance `list-⟨level⟩`, and the inner instance `description` (no dependency on the nesting level), i.e., the environment has the same appearance on all nesting levels.

### 3.7 The list environment

The generic `list` environment of  $\text{\LaTeX} 2_{\epsilon}$  is modeled with a `blockenv` instance named `list`, a `block` instance named `list- $\langle level \rangle$` , and an inner instance named `legacy` (with no dependency on the nesting level). This environment has two arguments and customization of the layout is expected to be directly set in the second argument. For this reason this `legacy` instance is something that shouldn't be changed (all that is attempted to provide a way to support legacy setups).

To set up the default settings (as they were used in  $\text{\LaTeX} 2_{\epsilon}$ ) the `setup-code` key gets `\legacylistsetupcode` assigned, so the default setup (that should probably not be changed) looks as follows:

```
\DeclareInstance{blockenv}{list}{display}
{
  name           = list,
  tag-name       = list,
  tagging-recipe = list,
  increment-level = true,
  setup-code     = \legacylistsetupcode ,
  block-instance = listblock ,
  inner-instance = legacy ,
}
```

### 3.8 The verse environment

The `verse` environment is currently still implemented as a list without real items (as in  $\text{\LaTeX} 2_{\epsilon}$ . That needs updating.

fix

### 3.9 The trivlist environment

In  $\text{\LaTeX} 2_{\epsilon}$  `trivlist` was used to define various display environments that aren't really lists at all. To support such legacy definitions (even though they should be updated to achieve proper tagging) we continue to support and implement it as a `list` environment with a few hardwired settings mimicking the original behavior.

### 3.10 Environments declared through `\newtheorem`

*to document*

## 4 Adjusting the layout of standard block environments

*to document*

## 5 Tagging support

### 5.1 Paragraph tags

Paragraphs in L<sup>A</sup>T<sub>E</sub>X can be nested, e.g., you can have a paragraph containing a display quote, which in turn consists of more than one (sub)paragraph, followed by some more text which all belongs to the same outer paragraph.

In the PDF model and in the HTML model that is not supported — a limitation that conflicts with real live, given that such constructs are quite normal in spoken and written language.

The approach we take to resolve this is to model such “big” paragraphs with a structure named `<text-unit>` and use `<text>` (rollmapped to `<P>`) only for (portions of) the actual paragraph text in a way that the `<text>`s are not nested. As a result we have for a simple paragraph the structures

```
<text-unit>
  <text>
    The paragraph text ...
  </text>
</text-unit>
```

The `<text-unit>` structure is rollmapped to `<Part>` or possibly to `<Div>` so we get a valid PDF, but processors who care can identify the complete paragraphs by looking for `<text-unit>` tags.

In the case of an element, such as a display quote or a display list inside the paragraph, we then have

```
<text-unit>
  <text>
    The paragraph text before the display element ...
  </text>
  <display element structure>
    Content of the display structure possibly involving inner <text-unit> tags
  </display element structure>
  <text>
    ... continuing the outer paragraph text
  </text>
</text-unit>
```

In other words such a display block is always embedded in a `<text-unit>` structure, possibly preceded by a `<text>...</text>` block and possibly followed by one, though both such blocks are optional.

Thus an `itemize` environment that has some introductory text but no text immediately following the list would be tagged as follows:

```
<text-unit>
  <text>
    The intro text for the itemize environment ...
  </text>
```

```

<itemize>
  <LI>
    <Lb1> label </Lb1>
    <LBody>
      The text of the first item involving <text-unit> as necessary ...
    </LBody>
  </LI>
  <LI>
    The second item ...
  </LI>
  ... further items ...
</itemize>
</text-unit>

```

The `<itemize>` is rollmapped to `<L>`.

For some display blocks, such as centered text, we use a simpler strategy. Such blocks still ensure that they are inside a `<text-unit>` structure but their body uses simple `<text>` blocks and not `<text-unit><text>` inside, e.g., the input

```

This is a paragraph with some
\begin{center}
  centered lines

  with a paragraph break between them
\end{center}
followed by some more text.

```

will be tagged as follows:

```

<text-unit>
  <text>
    This is a paragraph with some
  </text>
  <text /0 /Layout /TextAlign/Center>
    centered lines
  </text>
  <text /0 /Layout /TextAlign/Center>
    with a paragraph break between them
  </text>
  <text>
    followed by some more text.
</text-unit>

```

## 5.2 Tagging recipes

There are a number of different tagging recipes that implement different tagging approaches. They are selected through the `tagging-recipe` of the `blockenv` template. Currently the following values are implemented:

**standalone** This recipe does the following:

- Ensure that the `blockenv` is not inside a `<text-unit>` structure. If necessary, close the open one (and any open `<text>` structure).
- Text inside the body of the environment start with `<text-unit><text>` unless the key `tagging-suppress-paras` is set to `true` (which is most likely the wrong thing to do because we then get just `<text>` as the structure).
- At the end of the environment close `</text>` and possibly an inner `</text-unit>` if open.
- Finally, ensure that after the environment a new `<text-unit>` is started, if appropriate, e.g., if text is following.

**basic** This recipe does the following:

- Ensure that the `blockenv` is inside a `<text-unit>` structure, if necessary, start one.
- If inside a `<text-unit><text>`, then close the `</text>` but leave the `<text-unit>` open.
- Text inside the body of the environment start with `<text-unit><text>` if `tagging-suppress-paras` is set to `false`, otherwise just with `<text>`.
- At the end of the environment close `</text>` and possibly an inner `</text-unit>` if open.
- Then look if the environment is followed by an empty line (`\par`). If so, close the outer `</text-unit>` and start any following text with `<text-unit><text>`. Otherwise, don't and following text restarts with a just a `<text>` (and no paragraph indentation)

**standard** This recipe is like the **basic** one as far as handling `<text-unit>` and `<text>` is concerned. In addition

- it starts an inner tagging structure (i.e., which is therefore a child of the outer `<text-unit>`).
- By default this structure is a `<Div>` unless overwritten by the key `tag-name`. If that key is used, a suitable rolemap needs to be provided for the name given.
- At the end of the environment that inner structure is closed again so that we are back on the `<text-unit>` level from the outside.
- Then the lookahead for an empty line is done as described previously.

**list** This recipe is like the **standard** one except that

- the inner structure is a list (`<L>`).
- Furthermore everything is set up so that we have list items (`<LI>`) with suitable substructures (`<Lb1>` for the item labels and `<LBody>` for the item bodies).
- If the key `tag-name` is specified, this is used as the tag name for the whole list instead of `<L>`. Of course, it should then have a suitable rollmap.
- If the key `tag-attr-class` is specified then this is used as the class attribute. Again, this requires a suitable setup on the outside.

- At the end of the environment the `</LBody>`, `</LI>`, and `</L>` (or the tag name used) are closed.
- Then the lookahead for an empty line is done as described previously.

## 6 Debugging

---

```
\DebugBlocksOn
\DebugBlocksOff
\block_debug_on:
\block_debug_off:
```

---

These commands enable/disable debugging messages.

## 7 New and redefined kernel command

---

`\@doendpe` The original  $\text{\LaTeX} 2_{\epsilon}$  command is augmented to allow for tagging.

---

```
\legacyverbatimsetup
\legacylistsetupcode
```

---

*to be documented*

---

```
\@setupverbinvisiblepace
```

---

A counterpart definition to the kernel command `\@setupverbinvisiblepace`, needed as we need to handle real space chars in verbatim.

---

```
endblockenv
\g_block_nesting_depth_int
```

---

*to be documented*

---

```
\newtheorem
\@thm
\@begintheorem
```

---

Redefined to make theorems tagging aware.

---

```
\item
\@itemlabel
```

---

The `\item` is redefined.

---

```
\c@maxblocklevels
```

---

A counter to increase or decrease the number of supported level. If increased, one needs to supply additional level instances.

---

```
\begin
```

---

The `\begin` is slightly redefined to handle `\@doendpe` better. TODO: move to kernel

---

`\para_end`: TODO: consider name, document

---

`para/begin` The para/begin hook is enhanced to support list ends

## 8 The Implementation

```
1 <*package>
2 <@@=block>
3 \ProvidesPackage {latex-lab-testphase-block}
4     [\tlabblockdate\space v\tlabblockversion\space
5         blockenv implementation]
```

General kernel changes, also loaded by the sec and toc code.

```
6 \RequirePackage{latex-lab-kernel-changes}
```

For testing we temporarily load it here (it has to come before the definition of `\DebugBlocksOff` below:

```
7 \RequirePackage{latex-lab-testphase-context}
```

```
8 \ExplSyntaxOn
```

### 8.1 Handling `\par` after the end of the list

An empty line (or a `\par`) after a list has semantic meaning as it defines whether then following text is logically within the same paragraph as the list (no empty line) or whether it starts a new paragraph and the paragraph containing the list ends at the end of the list (empty line after the list). This is handled by  $\text{\LaTeX}$  using a legacy flag called `@endpe` and set of commands inside the generic `\end` (calling `\@doendpe`) and as part of the list environments identifying themselves as “paragraph ending environments” (by setting this flag).

For the reimplementaion of the list environments including support of tagging we need to augment that mechanism slightly and add some kernel hook(s) to add the tagging code if needed.

`\@doendpe` The original  $\text{\LaTeX}$   $2_{\epsilon}$  command is augmented to allow for tagging. TODO: use sockets for this and move to the kernel eventually.

```
9 \def\@doendpe{\@endpetrue
10 \def\par
11     {
```

If we are processing a  $\text{\$}$  math display and we encounter a real `\par` after it, we need to add a `\parskip` when tagging is done, because the one added by  $\text{\TeX}$  is always canceled by the processing in `\__math_tag_dollardollar_display_end`: in that case. This is signaled by the global legacy switch `@domathendpe` which is set to true in that case. Once the skip is applied we set it to false. If there is no `\par` at all, it will be reset in `\everypar` when the next paragraph starts.



```

12     \if@domathendpe
13         \skip_vertical:n { \tex_parskip:D }
14     \@domathendpefalse
15     \fi
16     \@restorepar
17     \clubpenalty\@clubpenalty

```

At this point we add the tagging code that closes an open `<text-unit>`, `<text>` tag combination, if necessary:

```

18     \tag_socket_use:n {@doendpe}

```

The standard `\par` command (`\par_end:`) acts on `@endpe` and attempts to close a still open `<text-unit>`s and this would be wrong if it was already closed above. So we have to reset the switch to false first.

```

19     \@endpefalse
20     \everypar{}
21     \par
22 }
23 \everypar{{\setbox\z@\lastbox}
24     \everypar{}
25     \@endpefalse

```

Not sure what is faster: testing for the status of the switch or setting it unconditionally to false (globally), probably roughly the same, so we set it always:

```

26 %     \if@domathendpe
27 %     \@domathendpefalse
28 %     \fi
29 }
30 }

```

*(End of definition for \@doendpe. This function is documented on page 15.)*

`tagsupport/@doendpe (socket)` The socket used in the `\@doendpe` TODO: if this goes into the kernel, the name should probably be different.

```

31 \socket_if_exist:nF{ tagsupport/@doendpe }
32 {
33     \NewTaggingSocket {@doendpe}{0}
34 }

```

`default (plug)` If a display block ends and is followed by a blank line we have to end the enclosing paragraph tagging structure.

```

35 \NewTaggingSocketPlug {@doendpe}{default}
36 {
37     \bool_if:NT \l__tag_para_bool
38     {

```

Given that restoring `\par` through the legacy L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> method can take a few iterations (for example, in case of nested lists, e.g., `... \end{itemize} \item ... \par` it can happen that the socket code is called while `@endpe` is already handled and then we should not attempt to close a `<text-unit>` structure). So we need to check for this.

```

39     \legacy_if:nT { @endpe }
40     {

```

If the display block currently ending was “flattened” (i.e., uses simplified paragraphs that are not tagged by a combination of `<text-unit>` followed by `<text>`, but simply with a `<text>`), then we don’t have to do anything, because the `<text>` is already closed.

```

41         \__block_debug_typeout:n
42         { flattened= \bool_if:NTF
43             \l__tag_para_flattened_bool
44             {true}{false}
45             \on@line }
46         \bool_if:NF \l__tag_para_flattened_bool
47         {
48             \__block_debug_typeout:n{Structure-end~
49             \l__tag_para_main_tag_tl\space
50             after~ displayblock \on@line }
51             \__tag_gincr_para_main_end_int:
52             \tag_struct_end: %text-unit
53         }
54     }
55 }
56 }
57 \AssignTaggingSocketPlug{@doendpe}{default}

```

```

\if@domathendpe Signal that special paragraph handling after a math display is required.
\@domathendpefalse
\@domathendpetrue
58 \newif\if@domathendpe
59 \def\@domathendpefalse{\global\let\if@domathendpe\iffalse}
60 \def\@domathendpetrue {\global\let\if@domathendpe\iftrue}

```

(End of definition for `\if@domathendpe`, `\@domathendpefalse`, and `\@domathendpetrue`.)

## 8.2 Template types and template interfaces

`blockenv` (*templatetype*) All template types expect a single key–value argument used to tweak template parameters specific to a given use in the document. This section is devoted to template interfaces, `block` (*templatetype*) `para` (*templatetype*) and the template code is covered later.

```

list (templatetype)
item (templatetype) \NewTemplateType{blockenv}{1}
62 \NewTemplateType{block}{1}
63 \NewTemplateType{para}{1}
64 \NewTemplateType{list}{1}
65 \NewTemplateType{item}{1}

```

`blockenv display` (*templ.*)

```

66 \DeclareTemplateInterface{blockenv}{display}{1}
67 {
68     name                : tokenlist ,

```

If not explicitly set then `tag-name` and `tag-attr-class` are set by the `tagging-recipe`. However, we have to default both to `<empty>` so that nested blocks do not inherit from the outer level.

```

69 tag-name           : tokenlist = ,
70 tag-attr-class     : tokenlist = ,
71 tagging-recipe     : tokenlist = standard,
72 increment-level    : boolean = true ,
73 setup-code         : tokenlist = ,
74 block-instance     : tokenlist = displayblock ,

```

Paragraph instance is normally inherited so no default.

```

75 para-instance      : tokenlist ,
76 inner-level-counter : tokenlist ,
77 max-inner-levels   : tokenlist = 4,
78 inner-instance-type : tokenlist = list ,
79 inner-instance     : tokenlist = ,
80 tagging-suppress-paras : boolean = false ,
81 final-code         : tokenlist = \ignorespaces ,
82 }

```

block display (*templ.*)

```

83 \DeclareTemplateInterface{block}{display}{1}
84 {
85   begin-vspace      : skip = \topsep ,
86   begin-extra-vspace : skip = \partopsep ,
87   para-vspace       : skip = \parsep ,
88   end-vspace        : skip = \KeyValue{begin-vspace} , % conflict with name below
89   end-extra-vspace  : skip = \KeyValue{begin-extra-vspace} ,
90   item-vspace       : skip = \itemsep ,
91   begin-penalty     : integer = \UseName{@beginparpenalty} ,
92   end-penalty       : integer = \UseName{@endparpenalty} ,
93   left-margin       : length = \leftmargin ,
94   right-margin      : length = \rightmargin ,
95   para-indent       : length = 0pt ,
96   % maybe add? (or more general for fonts and color)
97   % font             : tokenlist
98 }

```

para std (*templ.*)

```

99 \DeclareTemplateInterface{para}{std}{1}
100 {
101   para-attr-class   : tokenlist = justify ,
102   para-indent       : length = \parindent ,
103   begin-hspace      : skip = 0pt ,
104   left-hspace       : skip = 0pt ,
105   right-hspace      : skip = 0pt ,
106   end-hspace        : skip = \@flushglue ,
107   fixed-word-spaces : boolean = false ,
108   final-hyphen-demerits : integer = 5000 ,
109   newline-cmd       : tokenlist = \@normalcr ,
110 }

```

list std (*templ.*)

```

111 \DeclareTemplateInterface{list}{std}{1}
112 {
113   counter          : tokenlist = ,
114   item-label       : tokenlist = ,
115   start            : integer = 1 ,
116   resume           : boolean = false ,
117   item-instance    : instance{item} = basic ,
118   item-vspace      : skip = \itemsep ,
119   item-penalty     : integer = \UseName{@itempenalty} ,
120   item-indent      : length = \itemindent ,
121   label-width      : length = \labelwidth ,
122   label-sep        : length = \labelsep ,
123   legacy-support   : boolean = false ,
124 }

```

item std (*templ.*)

```

125 \DeclareTemplateInterface{item}{std}{1}
126 {
127   counter-label    : function{1} = \arabic{#1} ,
128   counter-ref      : function{1} = \KeyValue{counter-label} ,
129   label-ref        : function{1} = #1 ,
130   label-autoref    : function{1} = item~#1 ,
131   label-format     : function{1} = #1 ,
132   label-strut      : boolean = false ,
133   label-align      : choice {left,center,right,parleft} = right ,
134   label-boxed      : boolean = true ,
135   next-line        : boolean = false ,
136   text-font        : tokenlist ,
137   compatibility    : boolean = true ,
138 }

```

### 8.3 Useful helper commands

This section collects expl3 commands that will be useful.

`\_block_skip_set_to_last:N` Set a skip register to the value of an immediately preceding skip or zero if there was  
`\_block_skip_remove_last:` none.

```

139 \cs_new_protected:Npn \_block_skip_set_to_last:N #1 {
140   \skip_set:Nn #1 { \tex_lastskip:D }
141 }

```

Remove a skip previous skip if it is directly in front (not allowed in unrestricted vertical mode).

```

142 \cs_new_eq:NN \_block_skip_remove_last: \tex_unskip:D

```

(End of definition for `\_block_skip_set_to_last:N` and `\_block_skip_remove_last:.`)

```

143 \cs_generate_variant:Nn \tl_if_novalue:nTF { o }

```

### 8.3.1 Debugging

`\g__block_debug_bool`

```
144 \bool_new:N \g__block_debug_bool
```

*(End of definition for `\g__block_debug_bool`.)*

`\__block_debug:n`

`\__block_debug_typeout:n`

```
145 \cs_new_eq:NN \__block_debug:n \use_none:n
```

```
146 \cs_new_eq:NN \__block_debug_typeout:n \use_none:n
```

*(End of definition for `\__block_debug:n` and `\__block_debug_typeout:n`.)*

`\block_debug_on:`

`\block_debug_off:`

```
\__block_debug_gset #47 \cs_new_protected:Npn \block_debug_on:
```

```
148 {
```

```
149   \bool_gset_true:N \g__block_debug_bool
```

```
150   \__block_debug_gset:
```

```
151 }
```

```
152 \cs_new_protected:Npn \block_debug_off:
```

```
153 {
```

```
154   \bool_gset_false:N \g__block_debug_bool
```

```
155   \__block_debug_gset:
```

```
156 }
```

```
157 \cs_new_protected:Npn \__block_debug_gset:
```

```
158 {
```

```
159   \cs_gset_protected:Npx \__block_debug:n ##1
```

```
160   { \bool_if:NT \g__block_debug_bool {##1} }
```

```
161   \cs_gset_protected:Npx \__block_debug_typeout:n ##1
```

```
162   { \bool_if:NT \g__block_debug_bool { \typeout{[Blocks]~ ==>~ ##1} } }
```

```
163 }
```

*(End of definition for `\block_debug_on:`, `\block_debug_off:`, and `\__block_debug_gset:`. These functions are documented on page 15.)*

`\DebugBlocksOn` If we are debugging blocks we also want to know about template instances, so we turn  
`\DebugBlocksOff` the debugging for templates as well (for now).

```
164 \cs_new_protected:Npn \DebugBlocksOn { \block_debug_on: \template_debug_on: }
```

```
165 \cs_new_protected:Npn \DebugBlocksOff { \block_debug_off: \template_debug_off: }
```

```
166 \DebugBlocksOff
```

*(End of definition for `\DebugBlocksOn` and `\DebugBlocksOff`. These functions are documented on page 15.)*

## 8.4 Implementation of templates

### 8.4.1 Implementation of blockenv templates ...

`\g_block_nesting_depth_int` L<sup>A</sup>T<sub>E</sub>X<sub>2 $\epsilon$</sub>  already has a counter to record the nesting depth of blocks, but we want our own name because it isn't really tied to "lists" any more. However, `\@listdepth` is really part of the legacy interface (for example `minipage` alters it to point to a different counter) so that we are stuck with using at least indirectly for now and the following line makes this look like an L3 integer variable but internally expands to `\@listdepth`:

```
167 \cs_new:Npn \g_block_nesting_depth_int { \@listdepth } % a fake int
168                                         % for now
```

*(End of definition for `\g_block_nesting_depth_int`. This function is documented on page 15.)*

`blockenv display (templ.)`

```
169 \DeclareTemplateCode{blockenv}{display}{1}
170 {
171   name           = \l__block_env_name_tl ,
172   tag-name       = \l__block_tag_name_tl ,
173   tag-attr-class = \l__block_tag_class_tl ,
174   tagging-recipe = \l__block_tagging_recipe_tl ,
175   increment-level = \l__block_level_incr_bool ,
176   setup-code     = \l__block_setup_code_tl ,
177   block-instance = \l__block_block_instance_tl ,
178   para-instance  = \l__block_para_instance_tl ,
179   tagging-suppress-paras = \l__tag_para_flattened_bool ,
180   inner-level-counter = \l__block_inner_level_counter_tl ,
181   max-inner-levels = \l__block_max_inner_levels_tl ,
182   inner-instance-type = \l__block_inner_instance_type_tl ,
183   inner-instance  = \l__block_inner_instance_tl ,
184   final-code     = \l__block_final_code_tl ,
185 }
186 {
187   \__block_debug_typeout:n{\l__block_env_name_tl -env-start}
188 }
```

We first evaluate the key list passed from the document (if any). All known keys are used all, the remainder is stored in `\UnusedTemplateKeys` to be passed to any inner instances below.

```
189 \SetKnownTemplateKeys{blockenv}{display}{#1}
```

We need to know later if we have nested blockenvs inside a flattened environment. Whenever we start a new blockenv we increment `\l__tag_block_flattened_level_int` if it is already different from zero. If it is zero we increment it if flattening is requested. Thus a value of 0 means no flattening requested so far and 1 means this is the first blockenv requesting flattening. In either case we have to make sure that the blockenv is surrounded by a `<text-unit>` tag, while for any value above 1 we have to omit the `<text-unit>`.

```
190 \int_compare:nNnTF \l__tag_block_flattened_level_int > 0
191   {
```

```

192     \int_incr:N \l__tag_block_flattened_level_int
193   }
194   {
195     \bool_if:NT \l__tag_para_flattened_bool
196       {
197         \int_incr:N \l__tag_block_flattened_level_int
198       }
199   }

200 \tl_if_empty:NF \l__block_inner_level_counter_tl
201 {
202   \int_compare:nNnTF \l__block_inner_level_counter_tl >
203     { \l__block_max_inner_levels_tl - 1 }
204     { \@toodeep }
205     { \int_incr:N \l__block_inner_level_counter_tl } % not clean "o"?
206 }

```

Legacy defaults are only roped in if the list level changes. For display blocks that remain on the same level the current values are kept.

```

207 \bool_if:NT \l__block_level_incr_bool
208 {
209   \int_compare:nNnTF \g_block_nesting_depth_int >
210     { \c@maxblocklevels - 1 }
211     { \@toodeep }
212     {
213       \int_gincr:N \g_block_nesting_depth_int

```

If there are no legacy defaults for that level then the next line does nothing, i.e., the current values (from the last level become the defaults for the next.

```

214     \use:c { @list \int_to_roman:n
215             { \g_block_nesting_depth_int } }
216   }
217 }

```

If we are doing tagging we load one of the available recipes for tagging, which alters various kernel hooks to add appropriate tagging structures.

```

218 \UseTaggingSocket{block/recipe}{\l__block_tagging_recipe_tl}

```

The default for `list` environments is that they have an empty label and are not numbered (something that is then overwriting by the setup of a specific list). We ensure this here even for non-lists, because we need a defined state that then can be overwriting by the legacy setup code for the `list` environment in `\l__block_setup_code_tl`. This is needed in case lists are nested as they otherwise would inherit outer values (and suddenly an `itemize` would start incrementing an outer `enumerate` counter, etc.

```

219 \tl_clear:N \@itemlabel
220 \tl_clear:N \@listctr
221 \legacy_if_set_false:n { @nmbrrlist }

```

Then run the setup code if any is given in the instance.

```
222 \l__block_setup_code_tl
```

Next call a block instance at the appropriate level passing it any remaining key/value from the optional document-level argument (i.e., those now stored in `\UnusedTemplateKeys`).

```
223 \exp_args:Nee \UseInstance{block}
224     { \l__block_block_instance_tl - \int_use:N
225       \g_block_nesting_depth_int }
226     \UnusedTemplateKeys
```

After this instance has been processed, any remaining unused keys are stored in `\UnusedTemplateKeys` and we can make use of this data later as long as we do not call another instance that also does unused key processing and overwrites it. But this is what happens below, so we better save its current value for now.

```
227 \tl_set_eq:NN \l__block_unused_blockenv_keys_tl \UnusedTemplateKeys
```

After the block instance call the `para` and then `inner` (list) instance if either or both are specified (which may not be the case).

```
228 \tl_if_empty:NF \l__block_para_instance_tl
229 {
```

For now we don't offer to alter instance parameters here so we pass an empty argument.

```
230     \exp_args:Ne \UseInstance{para}{ \l__block_para_instance_tl } {}
231 }
```

The `inner` instance may have its own levels or none depending on which the instance name differs. Again we pass it the optional key/value list.

```
232 \tl_if_empty:NF \l__block_inner_instance_tl
233 {
234     \exp_args:Nee
235     \UseInstance{ \l__block_inner_instance_type_tl }
236     { \l__block_inner_instance_tl
237       \tl_if_empty:NF \l__block_inner_level_counter_tl
238         % not clean use "o"?
239         { - \int_use:N \l__block_inner_level_counter_tl }
240     }
241     \l__block_unused_blockenv_keys_tl
```

Again the instance may has processed a few keys from the sofar unused keys, so we update `\l__block_unused_blockenv_keys_tl` to match the new reality.

```
242     \tl_set_eq:NN \l__block_unused_blockenv_keys_tl \UnusedTemplateKeys
243 }
```



At this point, the `\l__block_unused_blockenv_keys_tl` token list should either be empty or it should contain only keys that are suitable for the item template, but right now there is no code to test that can test the latter; it would help probably if we have an interface for this.

**fix** For now we handle that when the first item is encountered, but that isn't really clean.

```
244 \tl_if_empty:NF \l__block_unused_blockenv_keys_tl
245 {
246   % check if only item template keys remain
247 }
```

We finish off with `\l__block_final_code_tl` which defaults to `\ignorespaces` so that spaces between `\begin{...}` and the start of the text are ignored.

```
248 \l__block_final_code_tl
249 }
```

`\l__block_unused_blockenv_keys_tl`

```
250 \tl_new:N \l__block_unused_blockenv_keys_tl
```

*(End of definition for `\l__block_unused_blockenv_keys_tl`.)*

`\l__tag_block_flattened_level_int` Count the levels of nested blockenvs starting with the first that is “flattened”. The counter is defined in `ltagging.dtx`, but until the next release 11/24 we set it up here too

```
251 \int_if_exist:NF \l__tag_block_flattened_level_int
252 {
253   \int_new:N \l__tag_block_flattened_level_int
254 }
```

*(End of definition for `\l__tag_block_flattened_level_int`.)*

**`\c@maxblocklevels`** A counter to increase or decrease the number of supported level. If increased, one needs to supply additional level instances.

```
255 \newcounter{maxblocklevels}
256 \setcounter{maxblocklevels}{6}
```

*(End of definition for `\c@maxblocklevels`. This function is documented on page 15.)*

**`\endblockenv`** The code executed when a blockenv ends is 99% the same for all blockenvs (at least up to now). Small differences exist, though. They are accounted for first in the conditionals.

We make this a public command so that new block environments can be set up without the need to resort to L3 layer programming.

**name is bad**

```
257 \cs_new:Npn \endblockenv {
258   \__block_debug_typeout:n{blockenv~ common~ ending \on@line}
```

If this block was incrementing the level we have to decrement it now again:

```
259 \bool_if:NT \l__block_level_incr_bool
260 { \int_gdecr:N \g_block_nesting_depth_int }
```

If this block was a list and there are still `\item` labels to be placed we move to horizontal mode to get them typeset.

```

261 \legacy_if:nT { @inlabel }
262   {
263     \mode_leave_vertical:
264     \legacy_if_gset_false:n { @inlabel }
265   }

```

If we are ending a list environment and we have not seen any `\item`, i.e., `@newlist` is still true, we raise an error. In basic a “displayblock” scenario `@newlist` will always be false, but if such an environment appears inside an outer list then `\noitemerr` could still be triggered and that is undesirable (as the missing item will be detected at the wrong point and again later, during the outer list processing). We therefore run it only if the current environment is a list.

```

266 \__block_if_list:T { \legacy_if:nT { @newlist } { \noitemerr } }

267 \mode_if_horizontal:TF
268   { \__block_skip_remove_last: \__block_skip_remove_last: \par }
269   { \inmatherr{\end{\currenvir}} }

```

Once we are back in vertical mode we can add the appropriate closing tagging structure(s), if we are doing tagging.

```

270 \__kernel_displayblock_end:

```

Resetting the `@newlist` switch is also only done if the current environment is a list and not unconditionally.

```

271 \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }

```

What to do in terms of vertical spacing in different situations is still somewhat open to debate, right now this is more or less implementing what L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> list environment have been doing.

some redesign/ex-  
tensions here?

```

272 % \__block_debug_typeout:n{@nparlist =
273 % \legacy_if:nTF { @nparlist }{true}{false}}
274 \legacy_if:nF { @nparlist }
275   {
276     \__block_skip_set_to_last:N \l_tmpa_skip
277     \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
278     {
279       \skip_vertical:n { - \l_tmpa_skip }
280       \skip_vertical:n { \l_tmpa_skip + \parskip - \outerparskip }
281     }
282     \addpenalty \@endparpenalty
283     \addvspace \l__block_topsepadd_skip

```

L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> triggered the paragraph handling after a list at this point here, i.e., only if the list didn’t start a paragraph. One can make a case for that, but it can be somewhat surprising to the user and there is a good argument that even such a list could be followed explanatory text that is part of the same paragraph and doesn’t start a new one.

decide which logic we  
want to use! If the old  
logic is used we need to  
close the text-unit our-  
selves in the true branch

```

284 % \legacy_if_gset_true:n { @endpe }
285 }

```

So this is for now always done. Probably `\l_block_topseppadd_skip` above should be added only if the paragraph ends here and not if it continues, so this need some further cleanup.

Finally, we have a socket that handles the `\par` handling after the block. Normally, we use it with the `on` plug (check for a following `\par`) but in the case of standalone environments we assign it the `off` plug.

```

286 \socket_use:n {block/endpe}
287 }

```

*(End of definition for `\endblockenv`. This function is documented on page 15.)*

decide

revisit and correct

`\_block_if_list:T` The following code may need some redesigning, as there is no good test for “is this environment a ‘list’ that has `\items`”. For now this here does the trick well enough.

```

288 \cs_new:Npn \_block_if_list:T
289   { \tl_if_eq:NnT \l_block_block_instance_tl {listblock} }

```

*(End of definition for `\_block_if_list:T`.)*

`\_kernel_displayblock_end:` The kernel hook for tagging at the end of the block.

```

290 \cs_new:Npn \_kernel_displayblock_end: {
291   \_block_debug_typeout:n{\detokenize{\_kernel_displayblock_end:}}
292 }

```

*(End of definition for `\_kernel_displayblock_end:.`)*

`block/endpe (socket)` This socket is responsible for the end environment `\par` handling. We define two plugs for it (`on` and `off`).

```

293 \socket_new:nn {block/endpe}{0}

```

`on (plug)` The plugs set the legacy `@endpe` switch. This must always happen because block environments with different settings can be nested and should not inherit the setting from the outer environment.

```

294 \socket_new_plug:nnn{block/endpe}{on}

```

We can’t use `\legacy_if_gset_true:n` because this is now doing more than setting the legacy switch

```

295           { \endpetrue }
296 \socket_new_plug:nnn{block/endpe}{off}
297           { \endpefalse }

```

```

298 \socket_assign_plug:nn{block/endpe}{on}

```

## 8.4.2 Implementation of para templates ...

para std (*templ.*)

```
299 \DeclareTemplateCode{para}{std}{1}
300 {
301   para-indent      = \parindent ,
302   begin-hspace     = \l__par_start_skip ,           % name??
303   left-hspace      = \leftskip ,
304   right-hspace     = \rightskip ,
305   end-hspace       = \parfillskip ,
306   fixed-word-spaces = \l__par_fixed_word_spaces_bool , % name??
307   final-hyphen-demerits = \finalhyphendemerits ,
308   newline-cmd      = \\ ,
309   para-attr-class  = \l__tag_para_attr_class_tl ,
310 }
311 {
312   \SetTemplateKeys{para}{std}{#1}
313   \skip_set:Nn \@rightskip \rightskip
314 }
```

## 8.4.3 Implementation of block templates ...

block display (*templ.*)

In contrast to the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> implementation we do not directly use `\listparindent` here but a private register of the template. The reason is that block template instances are also used outside of lists.

```
315 \DeclareTemplateCode{block}{display}{1}
316 {
317   begin-vspace      = \topsep ,
318   begin-extra-vspace = \partopsep ,
319   para-vspace       = \parsep ,
320   end-vspace        = \l__block_botsep_skip ,
321   end-extra-vspace  = \l__block_parbotsep_skip ,
322   item-vspace       = \itemsep ,
323   begin-penalty     = \@beginparpenalty ,
324   end-penalty       = \@endparpenalty ,
325   right-margin      = \rightmargin ,
326   left-margin       = \leftmargin ,
327   para-indent       = \l__block_parindent_dim ,
328 }
329 {
330   \SetKnownTemplateKeys{block}{display}{#1}
```

The code largely follows the logic of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>'s `trivlist` implementation as far as it applicable for the “display block” but coded using the L<sup>3</sup> programming layer. However, we keep all the legacy variables (e.g., `@noskipsec`) if there is some chance that they are set in classes or packages.

```
331   \legacy_if:nT { @noskipsec } { \mode_leave_vertical: }
332   \skip_set:Nn \l__block_topsepadd_skip { \topsep }
333   \mode_if_vertical:TF
```

```

334     {
335     \skip_add:Nn \l__block_topsepadd_skip { \partopsep }

```

At this point it is safe to add tagging structure(s) so we have a kernel-owned hook here for tagging. This is used to possibly start a paragraph structure (to surround the block, for example, in case of lists) and possibly do some other preparation for tagging the block.

```

336     \__kernel_displayblock_beginpar_vmode:
337     }
338     {

```

If we are in horizontal mode then the displayblock has to return to vertical mode now (after removing any immediately preceding skip or kern. But before we actually issue the `\par` we execute a kernel hook in which we can add tagging code. This hook is “weird” because by default it does nothing, but if tagging is wanted it takes an argument and grabs the following `\par` in order to put tagging code before and after the `\par`.

```

339     \__block_skip_remove_last: \__block_skip_remove_last:
340     \__kernel_displayblock_beginpar_hmode:w \par
341     }

```

Now we are back to legacy list implementation ...

```

342     \legacy_if:nTF { @inlabel }
343     {
344     \legacy_if_set_true:n { @noparitem }
345     \legacy_if_set_true:n { @noparlist }
346     }
347     {
348     \legacy_if:nT { @newlist } { \@noitemerr }
349     \legacy_if_set_false:n { @noparlist }
350     \skip_set_eq:NN \l__block_effective_top_skip \l__block_topsepadd_skip
351     }
352     \skip_add:Nn \l__block_effective_top_skip { \parskip }

```

Next lines set some paragraph defaults, any of them may get overwritten if there is a `para-`instance specified on the `blockenv` instance.

```

353     \skip_zero:N \leftskip
354     \skip_set_eq:NN \rightskip \@rightskip
355     \skip_set_eq:NN \parfillskip \@flushglue

```

The next lines establish a `parshape` which is retained across paragraphs by executing `\para_end:` within a group and thus reestablishing the `parshape` for the next paragraph again. In case a list got started `\par` is ignored until we have seen an `\item` (or we have executed `\par` one thousand times.

```

356     \int_zero:N \par@deathcycles
357     \@setpar
358     {
359     \legacy_if:nTF { @newlist }
360     {

```

```

361     \int_incr:N \par@deathcycles
362     \int_compare:nNnTF \par@deathcycles > { 1000 }
363       { \noitemerr
364         { \para_end: }
365       }
366     }
367   {
368     { \para_end: }
369   }
370 }
371 \skip_set_eq:NN \@outerparskip \parskip
372 \skip_set_eq:NN \parskip \parsep

373 \dim_set_eq:NN \parindent \l_block_parindent_dim
374 \dim_add:Nn \linewidth { - \rightmargin - \leftmargin }
375 \dim_add:Nn \@totalleftmargin { \leftmargin }
376 \tex_parshape:D 1 ~ \@totalleftmargin \linewidth

```

This is the point where we are ready to add the tagging structure for the block, e.g., an <L>, a <Figure> or some other structure.

```

377   \__kernel_displayblock_begin:

```

Finally, we have to output the vertical separation and penalty at the start of the block and make corrections for a change in `\parskip` and some other housekeeping, unless this block is inside a list and the list `\item` has not yet placed. In that case the vertical space and penalty us suppressed. This is controlled through the legacy switches `@noparitem`, `minipage`, and `@nobreak`.

```

378   \legacy_if:nTF { @noparitem }
379     {
380       \legacy_if_set_false:n { @noparitem }
381       \hbox_gset:Nn \g_block_labels_box
382         {
383           \skip_horizontal:n { - \leftmargin }
384           \hbox_unpack_drop:N \g_block_labels_box
385           \skip_horizontal:n { \leftmargin }
386         }
387
388       \legacy_if:nF { @minipage } % Why this chunk of code?
389         {
390           \__block_skip_set_to_last:N \l_block_tmpa_skip
391           \skip_vertical:n { - \l_block_tmpa_skip }
392           \skip_vertical:n { \l_block_tmpa_skip +
393             \@outerparskip - \parskip }
394         }
395       {
396         \legacy_if:nTF { @nobreak }
397           { \addvspace{\skip_eval:n{\@outerparskip-\parskip}} }
398         {

```

document 2e  
logic used here

```

399         \addpenalty \@beginparpenalty
400         \addvspace \l_block_effective_top_skip
401         \addvspace{-\parskip}
402     }
403 }
404 }

```

Extra keys to support enumitem conventions:

```

405 \keys_define:nn { template/block/display }
406 {
407   ,topsep      .skip_set:N = \topsep
408   ,partopsep   .skip_set:N = \partopsep
409   ,listparindent .skip_set:N = \listparindent
410 }

```

```

\__kernel_displayblock_begin: The internal kernel hooks for tagging.
\__kernel_displayblock_beginpar_hmode:w
\__kernel_displayblock_beginpar_vmode:w
411 \cs_new:Npn \__kernel_displayblock_begin: {
412   \__block_debug_typeout:n
413   {\detokenize{\__kernel_displayblock_begin:}}
414 }

415 \cs_new:Npn \__kernel_displayblock_beginpar_hmode:w {
416   \__block_debug_typeout:n
417   {\detokenize{\__kernel_displayblock_beginpar_hmode:w}}
418 }

419 \cs_new:Npn \__kernel_displayblock_beginpar_vmode: {
420   \__block_debug_typeout:n
421   {\detokenize{\__kernel_displayblock_beginpar_vmode:}}
422 }

```

*(End of definition for \\_\_kernel\_displayblock\_begin:, \\_\_kernel\_displayblock\_beginpar\_hmode:w, and \\_\_kernel\_displayblock\_beginpar\_vmode:.)*

#### 8.4.4 Implementation of list templates ...

**\@itemlabel** Both \@itemlabel and \@listctr from the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> list implementation are used (or set) by various packages. We therefore use them too, so that these packages have a fighting chance to work with the new tagging-aware implementation for list.

```

423 \tl_new:N \@itemlabel      % should have a top-level definition
424 \tl_new:N \@listctr       % should have a top-level definition

```

*(End of definition for \@itemlabel and \@listctr. These functions are documented on page 15.)*

**\\_\_block\_evaluate\_saved\_user\_keys:nn** Keys set on individual list environments may be intended to alter the behavior of the template instance that defines the \item command. If meant to alter only a single \item command one would specify them in the optional argument of the \item, but if they should alter all items the right place would be the list environment. For this reason we need to store the values and then set them inside the \item template code using \SetKnownTemplateKeys in the appropriate context (template type and template

name). This is done in `\__block_evaluate_saved_user_keys:nn`. The context is provided in the two arguments (because different list environments may use different `\item` instances based on different templates. By default the command does nothing because most environments do not have user key settings.

```
425 \cs_new_eq:NN \__block_evaluate_saved_user_keys:nn \use_none:nn
```

Maybe something like this should become a public function, but for now this is a one-off for the `\item` command and therefore coded inline and internal to the block code.

```
426 %\cs_new:Npn \__block_save_user_keys:n #1 {
427 % \tl_if_empty:nTF {#1}
428 %   { \cs_set_eq:NN \__block_evaluate_saved_user_keys:nn \use_none:nn }
429 %   { \cs_set:Npe \__block_evaluate_saved_user_keys:nn ##1##2
430 %     { \SetKnownTemplateKeys{##1}{##2}{ \exp_not:n{#1} } }
431 %}
```

*(End of definition for `\__block_evaluate_saved_user_keys:nn`.)*

#### `list std (templ.)`

This template implements numbered and unnumbered lists and can be combined with display blocks or with inline blocks.

```
432 \DeclareTemplateCode{list}{std}{1}
433 {
434   counter          = \l__block_counter_tl,
435   item-label       = \l__block_item_label_tl,
436   start            = \l__block_counter_start_int ,
437   resume           = \l__block_resume_bool ,
438   item-instance    = \__block_item_instance:n ,
439   item-vspace      = \itemsep ,
440 % item-para-vspace = \parsep ,
441   item-penalty     = \@itempenalty ,
442   item-indent      = \itemindent ,
443   label-width      = \labelwidth ,
444   label-sep        = \labelsep ,
445   legacy-support   = \l__block_legacy_support_bool , % FMI questionable
446 }
447 {
448   \__block_debug_typeout:n{template:list:std}
449 %}
```

We start by looking at the user supplied keys in #1. If there aren't any we reset `\__block_evaluate_saved_user_keys:nn` to do nothing. Otherwise we evaluate and set the keys in the context of the current list template. In addition we prepare `\__block_evaluate_saved_user_keys:nn` for execution in the template for `\item`.

```
450 \tl_if_empty:oTF {#1}
451   { \cs_set_eq:NN \__block_evaluate_saved_user_keys:nn \use_none:nn }
452   {
453     \SetKnownTemplateKeys{list}{std}{#1}
```



The setup for `\__block_evaluate_saved_user_keys:nn` is a bit tricky and has to be done with `\cs_set:Npe` even though we don't want to expand anything and therefore use `\exp_not:n` inside. All this does is that any `#` passed in via `#1` is doubled (e.g., from `label-format=\fbox{#1}` which is represented as `... \fbox{##1}`). Otherwise, we would end up with a replacement text like

```
\SetTemplateKeys {#1}{#2}{label-format=\fbox {#1}}
```

instead of

```
\SetTemplateKeys {#1}{#2}{label-format=\fbox {##1}}
```

resulting in very odd and puzzling behavior later on.

```
454     \cs_set:Npe \__block_evaluate_saved_user_keys:nn ##1##2
455       { \SetKnownTemplateKeys{##1}{##2}{
456         \exp_not:o { \UnusedTemplateKeys }
457       }
458
459       \exp_not:n {
460         \tl_if_empty:NF \UnusedTemplateKeys
461         {
462           \msg_error:nnee { block } { unknown-keys }
463           { \l__block_env_name_tl \space environment}
464           \UnusedTemplateKeys
465         }
466       }
467     }
```

Has this list a counter name defined in the instance?

```
468   \tl_if_empty:NTF \l__block_counter_tl
469     {
```

If no counter name has been specified as part of the instance setup the list might still be numbered if it is a legacy list that uses `\usecounter` in the second argument of the legacy `list` environment. However, in that case we don't have to do much because `\usecounter` sets up `\@listctr` and sets it to zero so that the first item is numbered 1.

So all we do is to check if there was a `start` value given that differs from 1 and if so we change the counter value to match that. This makes it possible to define a legacy `list` in which the counter doesn't start with 1 by explicitly setting the counter value in the second argument of the `list` environment but also overwriting that through a `start` key setting on invocation.

```
470     \int_compare:nNnF \l__block_counter_start_int = 1
471     {
472       \int_gset:cn{ c@ \@listctr }
473       { \l__block_counter_start_int - 1 }
474     }
475   }
```

In that case we only check if we should resume a previous list (`\@listctr` should be set in that case through the legacy method as well so we should be able to use it).

If a counter is set in the list instance we use that one. This should be the name of a  $\LaTeX$  counter that is already allocated externally—no runtime check is made for this: if it is not declared one will get “no such counter” error when the list is used.

```

476 {
477   \@nmbrlisttrue
478   \tl_set_eq:NN \@listctr \l__block_counter_tl
479   \bool_if:NF \l__block_resume_bool
480     {
481       \int_gset:cn{ c@ \@listctr }
482       { \l__block_counter_start_int - 1 }
483     }
484 }

```

Does the current instance has an item label representation? This would be possible whether or not we have a numbered list. If yes, then we use this for `\@itemlabel`, otherwise we expect that `\@itemlabel` is provided from the outside, e.g., as part of the `list` environment argument.

```

485 \tl_if_empty:NF \l__block_item_label_tl
486 {
487   \tl_set_eq:NN \@itemlabel \l__block_item_label_tl
488 }

```

Finally, we signal that we are at the start of a new list (which affects how the first `\item` is handled and how `\par` commands are interpreted).

```

489 \legacy_if_gset_true:n { @newlist }

```

If we encounter horizontal material before the first `\item` we do want a `\@noitemerr` straight away, because afterwards we end up with tagging structure faults whose cause is the missing `\item`. So we setup up `\__block_item_everypar`: to test for this; when the first `\item` is encountered this will get reset. This is only relevant for vertical lists, when dealing with inline lists one would need to test for something else to identify that there is horizontal material between the start of the list and the first `\item` (maybe some `\spacefactor` trick could be used then, or the material is boxed first and the width is inspected as suggested by Joseph).

```

490 \cs_set_eq:NN \__block_item_everypar: \__block_item_everypar_first:
491 \__block_debug_typeout:n{template:list:std-end}
492 }

```

The message that is used above when we are left with keys that are unknown:

```

493 \msg_new:nnnn { block } { unknown-keys }
494 { Some~ keys~ specified~ on~ the~ #1~ are~ unknown. }
495 {
496   The~ following~ keys~ are~ unknown~ and~ their~

```

Think about a better implementation at some point.

```

497 values~ are~ ignored:\\
498 \space\space #2\\
499 Perhaps~ a~ misspelling~ or~ the~ current~ template~
500 instance~ uses~ special~ keys.
501 }

```

Extra keys to support enumitem conventions:

```

502 \keys_define:nn { template/list/std }
503 {
504   ,nosep .code:n =
505     \dim_zero:N \itemsep
506     \dim_zero:N \parsep
507     \dim_zero:N \topsep
508     \dim_zero:N \l_block_botsep_skip
509     \dim_zero:N \l_block_parbotsep_skip
510   ,midsep .skip_set:N = \topsep
511 }

```

#### 8.4.5 Implementation of \item template(s)

`item std (templ.)` The item template has one hidden key `label` which is not available on the template for setting because it is only used to receive any optional data passed to the `\item` command. We therefore declare it with `\keys_define:nn` and ensure that the optional argument data to `\item` (if it is not a key/value list already) is passed to this `label` key.

```

512 \keys_define:nn { template/item/std }
513   { label .tl_set:N = \l_block_label_given_tl }
514 \DeclareTemplateCode{item}{std}{1}
515 {
516   counter-label = \l_block_counter_label:n ,
517   counter-ref   = \l_block_counter_ref:n ,
518
519   label-ref     = \l_block_label_ref:n ,
520   label-autoref = \l_block_label_autoref:n ,
521   label-format  = \l_block_label_format:n ,
522   label-strut   = \l_block_label_strut_bool ,
523   label-boxed   = \l_block_label_boxed_bool ,
524   next-line     = \l_block_next_line_bool ,
525   text-font     = \l_block_text_font_tl ,
526   compatibility = \l_block_item_compatibility_bool ,

```

alignment is mostly wrong (test short medium and multiline labels)

next set of key not yet used

complete — This probably needs a different implementation (and needs completing)

```

526   label-align = {
527     left  = \tl_set:Nn \l_block_item_align_tl { \relax \hss } ,
528     center = \tl_set:Nn \l_block_item_align_tl { \hss \hss } ,
529     right = \tl_set:Nn \l_block_item_align_tl { \hss \relax } ,
530     parleft = \NOT_IMPLEMENTED ,
531   } ,
532 }

```

Then typeset the label at its natural width by applying `\__block_make_label_box:n` to the label given or to a label constructed from the counter. If it is boxed and reasonably short, add padding to make it at least of size `\labelwidth`, then add another layer of box. This way, when we unpack it in `\g__block_labels_box` it correctly remains boxed in those cases. Afterwards, in the `nextline` case add `\newline` if the label did not fit in the allotted space.

```
533 {
534   \__block_debug_typeout:n{template:item:std}
```

First deal with the key–value input, which in particular may provide a value for the label (the usual optional argument of `\item`). For this we set `\l__block_label_given_tl` to `\c_novalue_tl` so that we can identify if an optional argument was given.

```
535   \tl_set_eq:NN \l__block_label_given_tl \c_novalue_tl
```

First we evaluate and set any keys specified on the list environment by calling `\__block_evaluate_saved_user_keys:nn`. Then we do the same with all keys specified on this `\item` command (which may overwrite one or the other setting just made).

```
536   \__block_evaluate_saved_user_keys:nn {item}{std}
```

We don't care whether all of the user keys from the list level have been applied, but those explicitly set on the `\item` command should be applicable, so we generate an error if that isn't the case:

```
537   \SetKnownTemplateKeys{item}{std}{#1}
538   \tl_if_empty:NF \UnusedTemplateKeys
539     {
540     \msg_error:nnee { block } { unknown-keys }
541       { \noexpand\item command }
542       \UnusedTemplateKeys
543     }
```

If no optional argument was given then `\l__block_label_given_tl` is still equal to `\c_novalue_tl` and so we can distinguish that from `\item[]`.

```
544   \tl_if_novalue:oTF \l__block_label_given_tl
545     {
```

The rest of the code for this template needs work and is both incomplete and partly wrong.

```
546     \tl_if_blank:oF \@listctr { \@kernel@refstepcounter \@listctr }
547     \bool_if:NTF \l__block_item_compatibility_bool   % not sure that
548                                                         % conditional
549                                                         % makes sense
550     { \__block_make_label_box:n { \MakeLinkTarget[\@listctr]}%
551       \@itemlabel } } % TODO ?
552     { \__block_make_label_box:n { \MakeLinkTarget[\@listctr]}%
553       \__block_counter_label:n { \@listctr } } }
554   }
```

fix

```

555     {
556       \__block_debug_typeout:n{item~ with~ optional}
557       \__block_make_label_box:n {\MakeLinkTarget[\l__block_env_name_tl]{}\l__block_label_given}
558     \bool_if:nT
559       {
560         \l__block_label_boxed_bool
561 %   TODO: is \linewidth correct?
562         && \dim_compare_p:n
563           { \box_wd:N \l__block_one_label_box <= \linewidth }
564       }
565     {
566       \dim_compare:nNnT
567         { \box_wd:N \l__block_one_label_box } < \labelwidth
568         {
569           \hbox_set_to_wd:Nnn \l__block_one_label_box { \labelwidth }
570           {
571             \exp_after:wN \use_i:nn \l__block_item_align_tl

```

FMi: L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> keeps the label boxed inside (not unboxed). This means that the content stays rigid and does not vary based on glue setting in the line with the label. There are cases where we do want the unboxed version (I think enumitem offers that in some cases too) but it should probably not be the default.

```

572 %           TODO: customize?
573 %           \hbox_unpack_drop:N \l__block_one_label_box
574           \box_use_drop:N \l__block_one_label_box
575           \exp_after:wN \use_ii:nn \l__block_item_align_tl
576         }
577     }

```

Add another box level to the label box:

```

578     \hbox_set:Nn \l__block_one_label_box
579       { \box_use_drop:N \l__block_one_label_box }
580   }
581   \dim_compare:nNnTF { \box_wd:N \l__block_one_label_box } > \labelwidth
582     { \bool_set_true:N \l__block_long_label_bool }
583     { \bool_set_false:N \l__block_long_label_bool }
584   \hbox_gset:Nn \g__block_labels_box
585     {
586       \hbox_unpack_drop:N \g__block_labels_box
587       \skip_horizontal:n { \itemindent - \labelsep - \labelwidth }
588       \hbox_unpack_drop:N \l__block_one_label_box
589       \skip_horizontal:n { \labelsep }
590       \bool_if:NT \l__block_next_line_bool
591         { \bool_if:NT \l__block_long_label_bool { \nobreak \hfil \break } }
592       % version of \newline inside an hbox that will be unpacked
593     }
594   % TODO??? FMi what's that?
595   % \skip_set_eq:NN \parsep \l__block_item_parsep_skip

```

The next setting is for compatibility: The list template sets `\listparindent` to zero and otherwise doesn't use it any more. However, in the second argument of a legacy `list`

environment the user may have set it explicitly to some other value and whatever value it had was then used for `\parindent` within the list. Now we use its value only if it differs from zero but otherwise use whatever the template instances specify. This gives 99.9% compatibility for legacy documents. 100% for definitions using the `list` environment and a setting inside, but if the user used `\listparindent` within the document, e.g., inside a `verse` environment there there is one case in which the setting is ignored, i.e., when it was set back to zero. That's a rather unlikely scenario, but it is not impossible. However, I couldn't think of an approach that circumvents such boundary cases.

```
596 \dim_compare:nNnF \listparindent = {0pt}
597   { \dim_set_eq:NN \parindent \listparindent }
```

Placing the list label(s) is done when the paragraph for the `\item` is started, which executes `\__block_item_everypar:` inside `para/begin`. By default this command does nothing, now we change it to attach the pending label or labels.

```
598 \cs_set_eq:NN \__block_item_everypar: \__block_item_everypar_std:
599 }
```

`\l__block_item_align_tl`

```
600 \tl_new:N \l__block_item_align_tl
    (End of definition for \l__block_item_align_tl.)
```

`\l__block_one_label_box` Each label is typeset in `\l__block_one_label_box` to be measured. Once this is ready, it is put (boxed or unboxed) in `\g__block_labels_box`, together with any pending labels (for the case where a list begins just after `\item`). This is an analogue of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>'s `\@labels`, but it is always unboxed before use, to support both boxed and unboxed labels.

```
601 \box_new:N \l__block_one_label_box
602 \box_new:N \g__block_labels_box
    (End of definition for \l__block_one_label_box and \g__block_labels_box.)
```

`\l__block_long_label_bool` Track whether the `\l__block_one_label_box` is larger than `\labelwidth`.

```
603 \bool_new:N \l__block_long_label_bool
    (End of definition for \l__block_long_label_bool.)
```

`\__block_make_label_box:n` Make one label, wrapped in `\__block_label_format:n`, with an appropriate `\strut` and possibly `\makeatletter` in compatibility mode (used for the `list` environment).

```
604 \cs_new_protected:Npn \__block_make_label_box:n #1
605   {
606     \hbox_set:Nn \l__block_one_label_box
607     {
```

If we do tagging then the contents of this box may need to be wrapped into a structure, e.g., `<Lbl>`.

```

608     \tag_socket_use:nmn {block/list/label}{-}
609     {
610
611         \__block_label_format:n
612         {
613             \bool_if:NT \l__block_label_strut_bool { \strut }
614             \bool_if:NTF \l__block_legacy_support_bool
615                 \makelabel
616                 \use:n
617                 {#1}
618         }
619     }

```

And what gets opened also needs closing:

```

618     }
619 }
620 }

```

*(End of definition for \\_\_block\_make\_label\_box:n and \\_\_block\_label\_format:e.)*

**block/list/label** (*socket*) A tagging socket to tag the label. It takes two arguments so that it can transparently pass the label content. Declaration is in `ltagging`.

```

621 \socket_if_exist:nF {tagsupport/block/list/label}
622 {
623     \NewTaggingSocket{block/list/label}{2}
624 }

```

**default** (*plug*)

```

625 \def\LBody{LBody}
626 \NewTaggingSocketPlug{block/list/label}{default}
627 {
628     %
629     % FMI: this needs a different logic to decide when to make the label
630     %     an artifact (after cleaning up the \item code ), therefore
631     %     disabled for now
632     % \tl_if_empty:oTF \@itemlabel
633     % {
634     %     \tag_mc_begin:n {artifact}
635     % }
636     % {
637     \tagstructbegin{tag=Lbl}
638     \tagmcbegin{tag=Lbl}
639     % }
640     #2
641     \tagmcbend                                     % end mc-Lbl or artifact
642     % FMI: unconditionally for now
643     % \tl_if_empty:oF \@itemlabel
644     \tagstructend % end Lbl
645     \tagstructbegin{tag=LBody}
646 }
647 \AssignTaggingSocketPlug{block/list/label}{default}

```

`\_block\_item\_everypar:` The `\_block\_item\_everypar:` command is executed as part of `para/begin` but most of the time does nothing, i.e., it has the following default definition outside of lists (and `\_block\_item\_everypar\_first:` most of the time within lists).

```
648 \cs_new_eq:NN \_block\_item\_everypar: \prg\_do\_nothing:
```

```
649 \AddToHook{para/begin}[items]{\_block\_item\_everypar:}
```

Note that we have to make sure that the above code is executed after the hook chunk from `tagpdf` because the latter uses `@inlabel` to make a decision.

By the end of the day both should probably move into the kernel hook instead or, better, into sockets.

```
650 \DeclareHookRule{para/begin}{items}{after}{tagpdf}
```

What follows is the version that resets various legacy booleans and puts the label box in the right place and finally resets itself to do nothing next time. `\_block\_item\_everypar:` is set to this by the item template so that the next paragraph start runs the code below.

```
651 \cs_new_protected:Npn \_block\_item\_everypar\_std: {
652   \_block\_debug\_typeout:n{item~ everypar \on@line }
653   \legacy\_if\_set\_false:n { @minipage }
654   \legacy\_if\_gset\_false:n { @newlist }
655   \legacy\_if:nT { @inlabel }
656   {
657     \legacy\_if\_gset\_false:n { @inlabel }

658     \box\_if\_empty:NT \g\_para\_indent\_box { \kern - \itemindent }
659     \para\_omit\_indent:

660     \box\_use\_drop:N \g\_block\_labels\_box
```

After the labels are placed we start a paragraph structure (if appropriate). This is handled in the following kernel hook:

```
661   \_kernel\_list\_label\_after:

662   \penalty \c\_zero\_int
663   }
664   \legacy\_if:nTF { @nobreak }
665   {
666     \legacy\_if\_gset\_false:n { @nobreak }
667     \int\_set:Nn \clubpenalty { 10000 }
668   }
669   {
670     \int\_set\_eq:NN \clubpenalty \@clubpenalty
```

Once the label(s) are typeset and we are past any special `@nobreak` handling we reset `\_block\_item\_everypar:` to do nothing.



```

671     \cs_set_eq:NN \__block_item_everypar: \prg_do_nothing:
672     }
673 }

```

This is the definition of `\__block_item_everypar:` before the first `\item` is encountered.

```

674 \cs_new:Npn \__block_item_everypar_first: {
675   \legacy_if:nT { @newlist } { \noitemerr }
676 }

```

*(End of definition for `\__block_item_everypar:`, `\__block_item_everypar_std:`, and `\__block_item_everypar_first:`.)*

`\l_block_tmpa_skip`

```

677 \skip_new:N \l_block_tmpa_skip

```

*(End of definition for `\l_block_tmpa_skip`.)*

`\l_block_topsepadd_skip` `\l_block_effective_top_skip` Variables equivalent to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>'s `\@topsepadd` and `\@topsep`. Roughly equal to a mixture of `topsep`, `partopsep`, and various `parskip` at different nesting levels in lists. The code is really elaborate when `@inlabel` is true.

```

678 \skip_new:N \l_block_topsepadd_skip
679 \skip_new:N \l_block_effective_top_skip

```

*(End of definition for `\l_block_topsepadd_skip` and `\l_block_effective_top_skip`.)*

**\item** Here we already have all the building blocks. Complain in math mode. Distinguish between first item (do necessary tagging) and later items `\__block_inter_item:` to cleanly close what's before, then call `\__block_item_instance:n` (which calls `\UseInstance{item}{(instance)}`) to prepare the upcoming item: it will be actually inserted only once some later material triggers `\everypar`.

```

680 \AddToHook{begindocument/before}{
681   \RenewDocumentCommand{\item}{ = {label} o }
682   {
683     \@inmatherr \item

```

TODO: Check if test for being outside of a list is sensible

```

684   \cs_if_free:NTF \__block_item_instance:n
685   {
686     \@latex@error{Lonely~\string\item--perhaps~a~missing~
687     list~environment}\@ehc
688   }
689   {
690     \legacy_if:nTF { @newlist }
691     {
692       \__kernel_list_item_begin:

```

The first item of a list also has to change the `@newlist` switch.

```

693         \legacy_if_gset_false:n { @newlist }
694     }
695     { \__block_inter_item: }

```

To avoid unnecessary key/val processing we make a quick check if there was an optional argument.

```

696         \tl_if_novalue:nTF {#1}          % avoids reparsing label={ }
697         { \__block_item_instance:n { } }
698         { \__block_item_instance:n {#1} }

```

Set the legacy switch that signals that we have a pending item label:

```

699         \legacy_if_gset_true:n { @inlabel }
700         \ignorespaces
701     }
702 }
703 }

```

*(End of definition for \item. This function is documented on page 15.)*

`\__block_inter_item:` Between items. If the previous item had no content then we need to trigger `\everypar`. Otherwise we simply close the previous item with `\par` after removing some horizontal space. Between items, there is a penalty and some space.

```

704 \cs_new_protected:Npn \__block_inter_item: {
705     \legacy_if:nT { @inlabel }
706         { \indent \par } % case of \item\item

```

`\par` may have a strange definition and may not get us back to vertical mode in one go, so we better do not treat the next line as an else case to the above conditional (for now).

```

707     \mode_if_horizontal:T { \__block_skip_remove_last:
708         \__block_skip_remove_last: \par }

```

End any LI-tag, then start the next LI-tag (if doing tagging):

```

709     \__kernel_list_item_end:
710     \__kernel_list_item_begin:

711     \addpenalty \@itempenalty
712     \addvspace \itemsep
713 }

```

*(End of definition for \\_\_block\_inter\_item:.)*

`\__kernel_list_item_begin:`  
`\__kernel_list_item_end:`

```

714 \cs_new_eq:NN \__kernel_list_item_begin: \prg_do_nothing:
715 \cs_new_eq:NN \__kernel_list_item_end: \prg_do_nothing:

```

*(End of definition for \\_\_kernel\_list\_item\_begin: and \\_\_kernel\_list\_item\_end:.)*

## 8.5 Tagging support commands

In this section we provide code to the various kernel hooks to support the tagging of different displayblock environments.

`\_block\_beginpar\_vmode:` When a block starts out in vertical mode, i.e., is not yet part of a paragraph, we have to start a paragraph structure. However, this is not the case if we are already flattening paragraphs, thus in this case we do nothing. We also do nothing if `@endpe` is currently true, because that means we are right now just after the end of a `blockenv` and in the process of looking if we have to end the current `<text-unit>`, i.e., it is already open.

The command is mapped to `\_kernel\_displayblock\_beginpar\_vmode` in various tagging recipes. It is also used in the math code!

```

716 \cs_set:Npn \_block\_beginpar\_vmode: {
717   \_block\_debug\_typeout:n
718   { @endpe = \legacy\_if:nTF { @endpe }{true}{false} \on@line }
719   \legacy\_if:nTF { @endpe }
720   {
721     \legacy\_if\_gset\_false:n { @endpe }
722   }

```

We test for `<2` because the first flattened environment has to surround itself with a `<text-unit>`. Only any inner ones then have to avoid adding another `<text-unit>`.

```

723   {
724     \int\_compare:nNnT \l\_tag\_block\_flattened\_level\_int < 2
725     {
726       \_tag\_gincr\_para\_main\_begin\_int:
727       \tag\_struct\_begin:n
728       {
729         tag=\l\_tag\_para\_main\_tag\_tl,
730         attribute\_class=\l\_tag\_para\_main\_attr\_class\_tl,
731       }
732       \_tag\_para\_main\_store\_struct:
733     }
734   }
735 }

```

*(End of definition for `\_block\_beginpar\_vmode:.`)*

`\_block\_beginpar\_hmode:N` If the block is already part of a part of a paragraph, i.e., when it has some text directly in front, then the first thing to do is to return to vertical mode. However, that should be done without inserting a paragraph end tag, so before calling `\par` to do its normal work, we disable paragraph tagging and restarting afterwards again. The argument to this config point simply gobbles the `\par` following it in the code above (which is used when there is no tagging going on).

The command is mapped to `\_kernel\_displayblock\_beginpar\_hmode:w` in various tagging recipes.

```

736 \cs_set:Npn \_block\_beginpar\_hmode:N #1
737   {
738     \tag\_mc\_end:

```

```

739     \_tag_gincr_para_end_int:
740     \_block_debug_typeout:n{increment~ /P \on@line }
741     \bool_if:NT \l__tag_para_show_bool
742     { \tag_mc_begin:n{artifact}
743       \rlap{\color_select:n{red}\tiny\ \int_use:N\g__tag_para_end_int}
744       \tag_mc_end:
745     }
746     \tag_struct_end:
747     \tagpdfparaOff \par \tagpdfparaOn
748   }

```

(End of definition for \\_block\_beginpar\_hmode:N.)

Paragraph tagging is mainly done using the paragraph hooks. The code is currently in tagpdf and uses sockets. The default plug plain is not good enough when lists get supported: we need to delay starting the paragraph tagging if we still have to place the list label. tagpdf assigns therefore a different socket in the `package/latex-lab-testphase-block/after` hook. This should move into ltagging (and then perhaps use kernel para hooks).

`/block/startpara/direct` (*socket*) A tagging socket to start a paragraph structure. It takes an argument (which is only used in debugging) that should be gobbled if tagging is not active. Not yet in ltagging (name and function should be reviewed).

This is a similar code to the one used in the para/begin hook but without testing `@endpe`. This is not needed in the standalone case and wrong inside lists.

This code is used in various places and should be a dummy if tagging is not active.

```

749 \socket_if_exist:nF {tagsupport/block/startpara/direct}
750 {
751   \NewTaggingSocket {block/startpara/direct}{1}
752 }

```

default (*plug*)

```

753 \NewTaggingSocketPlug{block/startpara/direct}{default}
754 {
755   \bool_if:NF \l__tag_para_flattened_bool
756   {
757     \_tag_gincr_para_main_begin_int:
758     \tag_struct_begin:n
759     {
760       tag=\l__tag_para_main_tag_tl,
761       attribute-class=\l__tag_para_main_attr_class_tl,
762     }
763     \_tag_para_main_store_struct:
764   }
765   \_tag_gincr_para_begin_int:
766   \_block_debug_typeout:n{increment~ P \on@line }
767   \tag_struct_begin:n
768   {
769     tag=\l__tag_para_tag_tl
770     ,attribute-class=\l__tag_para_attr_class_tl

```

```

771     }
772     \__tag_check_para_begin_show:nn {green}{#1}
773     \tag_mc_begin:n {}
774 }
775 \AssignTaggingSocketPlug{block/startpara/direct}{default}

```

The para/end hook is currently handled by tagpdf which uses sockets there and assigns suitable sockets for the block code in a package/after hook. This code should move into ltagging and be active always. Currently we still need to remove the tagpdf chunk to avoid that the socket is added twice. We add empty chunks to avoid warning messages from code parts trying to remove the chunks.

```

776 \AddToHook{para/end}[tagpdf]{}
777 \RemoveFromHook{para/end}[tagpdf]
778 \AddToHook{para/end}{}
779 \def\PARALABEL{NP-}

```

port/kernel/endpe/vmode (socket) A tagging socket which ends a structure. Used in \begin and \para\_end: Not yet in ltagging (name and function should be reviewed).

```

780 \socket_if_exist:nF {tagsupport/kernel/endpe/vmode}
781 {
782     \NewTaggingSocket {kernel/endpe/vmode}{0}
783 }

```

default (plug)

```

784 \NewTaggingSocketPlug{kernel/endpe/vmode}{default}
785 {
786     \if@endpe \ifvmode
787         \bool_if:NT \l__tag_para_bool
788     {
789         \bool_if:NF \l__tag_para_flattened_bool
790         {
791             \__tag_gincr_para_main_end_int:
792             \tag_struct_end:
793         }

```

\@endpfalse is needed by \para\_end:, see test tagging-0097.

```

794     \@endpfalse
795 }
796     \fi \fi
797 }
798 \AssignTaggingSocketPlug{kernel/endpe/vmode}{default}

```

**\para\_end:** If we see a \par in vmode and a <text-unit> is still open we need to close that. For this we check if a request for @endpe was made (but the \par redefinition got lost due to (bad?) coding).

```

799 \cs_set_protected:Npn \para_end: {
800     \scan_stop:

```

```

801 \mode_if_horizontal:TF {
802   \mode_if_inner:F {
803     \tex_unskip:D
804     \hook_use:n{para/end}
805     \@kernel@after@para@end
806     \mode_if_horizontal:TF {
807       \if_int_compare:w 11 = \tex_lastnodetype:D
808         \tex_hskip:D \c_zero_dim
809       \fi:
810       \tex_par:D
811       \hook_use:n{para/after}
812       \@kernel@after@para@after
813     }
814     { \msg_error:nnnn { hooks }{ para-mode }{end}{horizontal} }
815   }
816 }
817 {

```

TODO 2025-07-01. This is not exactly as before, this doesn't insert an `\endpefalse` when tagging is active. Check if this a problem.

```

818   \UseTaggingSocket{kernel/endpe/vmode}%
819   \tex_par:D
820 }
821 }
822 \cs_set_eq:NN \par      \para_end:
823 \cs_set_eq:NN \_blockpar \para_end:
824 \cs_set_eq:NN \endgraf \para_end:

```

*(End of definition for `\para_end:`. This function is documented on page 16.)*

**`\begin`** We need to do a little more than canceling `@endpe` now.

```

825 \protected\def\begin#1{%
826   \UseHook{env/#1/before}%
827   \@ifundefined{#1}%
828     {\def\reserved@a{\@latex@error{Environment-#1~undefined}\@eha}}%
829     {\def\reserved@a{\def\@currenvir{#1}%
830       \edef\@currenvline{\on@line}%
831       \@execute@begin@hook{#1}%
832       \csname #1\endcsname}}%
833   \@ignorefalse
834   \begingroup
835   \UseTaggingSocket{kernel/endpe/vmode}%
836   \reserved@a}

```

*(End of definition for `\begin`. This function is documented on page 15.)*

`\_kernel_list_label_after:` If starting the text-unit/text tags got delayed because of a pending label we have to do it after the label got typeset. TODO: it should do nothing without tagging that's why there is a test, this should be better hidden in a tagging socket, but it is not quite clear how to do this.

```

837 \cs_new:Npn \__kernel_list_label_after: {
838   \bool_lazy_and:nnT { \tag_if_active_p: } {\l__tag_para_bool }
839   {
840     \tag_socket_use:nn {block/startpara/direct} { LI- }
841   }
842 }

```

*(End of definition for \\_\_kernel\_list\_label\_after:.)*

`\__block_inner_begin:` Start a block that has an inner structure if it isn't also a list. This command is tagging specific, it is mapped to `\__kernel_displayblock_begin:` in some tagging recipes.

```

843 \cs_new:Npn \__block_inner_begin: {
844   \tagstructbegin{tag=\l__block_tag_inner_tag_tl}
845 }

```

*(End of definition for \\_\_block\_inner\_begin:.)*

`\__block_inner_end:` End a block (which isn't also a list). This command is tagging specific, it is mapped to `\__kernel_displayblock_end:` in some tagging recipes.

```

846 \cs_new:Npn \__block_inner_end: {
847   \__block_debug_typeout:n{block-end \on@line}
848   \legacy_if:nT { @endpe }
849   {
850     \__tag_gincr_para_main_end_int:
851     \__block_debug_typeout:n{close~ /text-unit \on@line}
852     \tagstructend
853   }
854   \tagstructend      % end inner structure
855 }

```

*(End of definition for \\_\_block\_inner\_end:.)*

### 8.5.1 List tags

```

856 \tl_new:N \l__tag_L_tag_tl
857 \tl_set:Nn \l__tag_L_tag_tl {L}
858
859 \tl_new:N \l__tag_L_attr_class_tl
860 \tl_set:Nn \l__tag_L_attr_class_tl {list}

```

```

861 \tagpdfsetup
862 {
863   role/new-attribute = {itemize}
864                       {/0 /List /ListNumbering/Unordered},
865   role/new-attribute = {enumerate}
866                       {/0 /List /ListNumbering/Ordered},
867   role/new-attribute = {description}
868                       {/0 /List /ListNumbering/Description},

```

Initially, we had `/None` for the basic list environment, but that is not allowed in PDF/UA-2 if the list contains any `Lbl` tags. So now we default to `Unordered`.

```

869   role/new-attribute = {list}{/0 /List /ListNumbering/Unordered},

```

```
870 }
```

```
871 \def\LItag{LI}
```

`\_block\_list\_begin:` Start a list ... This command is tagging specific, it is mapped to `\_kernel\_displayblock\_begin:` in a tagging recipe.

```
872 \cs_set:Npn \_block\_list\_begin: {
873   \tagstructbegin
874   {
875     tag=\l\_tag\_L\_tag\_tl
876     ,attribute-class=\l\_tag\_L\_attr\_class\_tl
877   }
878 }
```

*(End of definition for \\_block\\_list\\_begin:.)*

`\_block\_list\_item\_begin:` Start tagging a list item. This command is tagging specific, it is mapped to `\_kernel\_list\_item\_begin:` in a tagging recipe.

```
879 \cs_set:Npn \_block\_list\_item\_begin: { \tagstructbegin{tag=\LItag} }
```

*(End of definition for \\_block\\_list\\_item\\_begin:.)*

`\_block\_list\_item\_end:` When a list item ends we have to close `<LBody>` and `<LI>` but also a `<text>` in the special case that the item material ends in a list (identifiable via `@endpe`). This command is tagging specific. This command is copied to `\_kernel\_list\_item\_end:` in the list recipe.

```
880 \cs_set:Npn \_block\_list\_item\_end: {
881   \legacy_if:nT { @endpe }
882   {
883     \_tag\_gincr\_para\_main\_end\_int:
884     \tagstructend % text-unit
885     % \_block\_debug\_typeout:n{Structure-end~ P~ at~ item-end \on@line }
886   }
887   \tagstructend \tagstructend % end LBody, LI
888 }
```

*(End of definition for \\_block\\_list\\_item\\_end:.)*

`\_block\_list\_end:` Finally, at the list end we have to close the open `<LBody>`, `<LI>`, `<L>`, and possibly a `<text>` if the last item ends with a list. However, if the user forgot to add an `\item` then there will be no `<LI>` and `<LBody>` open, so we check for the status of `@newlist`. The corresponding no-item error was generated earlier outside the tagging code.

One could argue that it doesn't matter if the tagging is wrong after a `\@noitemerr` was issued. However, there is one case where it isn't an error: In the `thebibliography` environment (which is internally a list) it is often the case that documents start out with an empty environment, not containing any `\bibitems`. For that reason `\@noitemerr` is redefined inside that environment to only produce a warning; hence we have to produce correct tag structures in that case. This command is tagging specific. This command is copied to `\_kernel\_displayblock\_end:` in the list recipe.

```
889 \cs_new:Npn \_block\_list\_end: {
```

If `@newlist` is true (i.e., when we have an error or warning situation) there is not much to close.



```

890 \legacy_if:nF { @newlist }
891 {
892   \legacy_if:nT { @endpe }
893   {
894     \__tag_gincr_para_main_end_int:
895     \tagstructend % text-unit
896     \__block_debug_typeout:n{Structure-end~ P~ at~ list-end \on@line }
897   }
898   \tagstructend\tagstructend % end LBody, LI
899 }
900 \tagstructend % end L
901 }

```

(End of definition for \\_\_block\_list\_end:.)

End of tagging related declarations.

## 8.6 Tagging recipes

`tagsupport/block/recipe (socket)` A tagging socket to call the tagging recipe. Declared in `ltagging`.

```

902 \socket_if_exist:nF {tagsupport/block/recipe}
903 {
904   \NewTaggingSocket{block/recipe}{1}
905 }

```

`default (plug)`

```

906 \NewTaggingSocketPlug{block/recipe}{default}
907 {
908   \use:c { __block_recipe_#1: }
909 }
910 \AssignTaggingSocketPlug{block/recipe}{default}

```

`\__block_recipe_basic:` The `basic` recipe simply ensures that the block is inside a `<text-unit>` structure and if necessary starts one. When the block ends and is followed by a blank line the `<text-unit>` structure is closed too, otherwise it remains open and further text starts with just a `<text>` structure.

There is otherwise no inner structure so `\__kernel_displayblock_begin:` and `\__kernel_displayblock_end:` do nothing—`blockenvs` with inner structure use the `standard` or `list` recipe instead.

```

911 \cs_new:Npn \__block_recipe_basic: {
912   \cs_set_eq:NN \__kernel_displayblock_beginpar_hmode:w
913   \__block_beginpar_hmode:N
914   \cs_set_eq:NN \__kernel_displayblock_beginpar_vmode:
915   \__block_beginpar_vmode:
916   \let \__kernel_displayblock_begin: \prg_do_nothing:
917   \let \__kernel_displayblock_end: \prg_do_nothing:

```

End environment `\par` handling:

```

918   \socket_assign_plug:nn{block/endpe}{on}
919 }

```

(End of definition for `\_block_recipe_basic:`.)

`\_block_recipe_standalone:`

The `standalone` recipe produces a block that ensures that a previous `<text-unit>` ends and that after the block a new `<text-unit>` starts.

```
920 \cs_new:Npn \_block_recipe_standalone: {
921   \cs_set_eq:NN \_kernel_displayblock_beginpar_hmode:w
922                                     \prg_do_nothing:
923   \cs_set_eq:NN \_kernel_displayblock_beginpar_vmode:
924                                     \prg_do_nothing:
925   \cs_set_eq:NN \_kernel_displayblock_begin: \_block_inner_begin:
926   \cs_set_eq:NN \_kernel_displayblock_end:   \_block_inner_end:
```

End environment `\par` handling:

```
927   \socket_assign_plug:nn{block/endpe}{off}

928   \tl_if_empty:NTF \l_block_tag_name_tl
929     { \tl_set:Nn \l_block_tag_inner_tag_tl {Sect}          }
930     { \tl_set_eq:NN \l_block_tag_inner_tag_tl \l_block_tag_name_tl }
931 }
```

(End of definition for `\_block_recipe_standalone:`.)

`\_block_recipe_standard:` The `standard` recipe does the following:

- surround the block with a `<text-unit>` structure if not already in a `<text-unit>`. In the latter case end the MC and the `<text>` but leave the `<text-unit>` open. If we are producing flattened paragraphs, just close any `<text>` but do not open a `<text-unit>`.
- Then open an new (inner) structure (by default `<Div>` but typically the one specified on the instance).
- At the end of the block close the inner structure (`<Div>` or explicit one) but leave the `<text-unit>` open to be either continued or closed due to a following `\par`.

```
932 \cs_new:Npn \_block_recipe_standard:
933 {
934   \cs_set_eq:NN \_kernel_displayblock_beginpar_hmode:w
935                                     \_block_beginpar_hmode:N
936   \cs_set_eq:NN \_kernel_displayblock_beginpar_vmode:
937                                     \_block_beginpar_vmode:
938   \cs_set_eq:NN \_kernel_displayblock_begin: \_block_inner_begin:
939   \cs_set_eq:NN \_kernel_displayblock_end:   \_block_inner_end:
```

End environment `\par` handling:

```
940   \socket_assign_plug:nn{block/endpe}{on}
```

```

941 \tl_if_empty:NTF \l__block_tag_name_tl
942   { \tl_set:Nn \l__block_tag_inner_tag_tl {Div} }
943   { \tl_set_eq:NN \l__block_tag_inner_tag_tl \l__block_tag_name_tl }
944 }

```

(End of definition for `\__block_recipe_standard:`.)

`\l__block_tag_inner_tag_tl` The tag name that is used if the block has an inner structure.

```

945 \tl_new:N \l__block_tag_inner_tag_tl

```

(End of definition for `\l__block_tag_inner_tag_tl`.)

`\__block_recipe_list:` The `list` recipe does the following.

- It opens a `<text-unit>`-structure or keeps the current one open (only closing the MC).
- It then starts a new structure rolemapped to L-structure and arranges for handling list items, e.g., Li, Lbl and LBody structures.
- At the end it closes open list structures as needed but keeps the `<text-unit>`-structure open to continue the paragraph after the list, if necessary.

```

946 \cs_new:Npn \__block_recipe_list:
947 {
948   \cs_set_eq:NN \__kernel_displayblock_beginpar_hmode:w
949                 \__block_beginpar_hmode:N
950   \cs_set_eq:NN \__kernel_displayblock_beginpar_vmode:
951                 \__block_beginpar_vmode:
952   \cs_set_eq:NN \__kernel_displayblock_begin: \__block_list_begin:
953   \cs_set_eq:NN \__kernel_displayblock_end: \__block_list_end:

```

The next two lines could be done globally, because they are only called if we do have `\items`, i.e., if we are in a list. It is therefore also not necessary to reset them in other recipes (right now—this may change if we get more templates (like inline lists)).

```

954   \cs_set_eq:NN \__kernel_list_item_begin: \__block_list_item_begin:
955   \cs_set_eq:NN \__kernel_list_item_end: \__block_list_item_end:

```

End environment `\par` handling:

```

956   \socket_assign_plug:nn{block/endpe}{on}

```

Handle the tag name and attribute classes using the key values from the current list instance.

```

957   \tl_if_empty:NTF \l__block_tag_name_tl
958     { \tl_set:Nn \l__tag_L_tag_tl {L} }
959     { \tl_set_eq:NN \l__tag_L_tag_tl \l__block_tag_name_tl }
960   \tl_if_empty:NTF \l__block_tag_class_tl
961     { \tl_set:Nn \l__tag_L_attr_class_tl {} }
962     { \tl_set_eq:NN \l__tag_L_attr_class_tl \l__block_tag_class_tl }
963 }

```

(End of definition for `\__block_recipe_list:`.)

## 9 Implementation of document-level block environments

Most such environments are pretty simple: they take an optional argument and call a `blockenv` instance to do the work. At the end of environment we call `\endblockenv` to finish.

### 9.1 Displayblock environments

There are two basic block environment which are similar to  $\text{\LaTeX} 2_{\epsilon}$ 's `trivlist` except that there aren't degenerated lists and thus have no hidden `\item` inside.

`displayblock (env.)`

```
964 \NewDocumentEnvironment{displayblock}{!0{ } }
965   { \UseInstance{blockenv}{displayblock} {#1} }
966   { \endblockenv }
```

`displayblockflattened (env.)`

```
967 \NewDocumentEnvironment{displayblockflattened}{!0{ } }
968   { \UseInstance{blockenv}{displayblockflattened} {#1} }
969   { \endblockenv }
```

### 9.2 The center, flushleft, and flushright environments

For now we redeclare various document environments as late as possible in order to make tagging work, even if classes have changed the definitions. Of course this means that such changes get lost.

```
970 \AddToHook{begindocument/before}{
```

`center (env.)`

`flushleft (env.)`

`flushright (env.)`

```
\RenewDocumentEnvironment{center} {!0{ } }
972   { \UseInstance{blockenv}{center}{#1} }
973   { \endblockenv }

974 \RenewDocumentEnvironment{flushright} {!0{ } }
975   { \UseInstance{blockenv}{flushright}{#1} }
976   { \endblockenv }

977 \RenewDocumentEnvironment{flushleft} {!0{ } }
978   { \UseInstance{blockenv}{flushleft}{#1} }
979   { \endblockenv }
980 }
```

### 9.3 Display quote environments

```
quote (env.)
quotation (env.)
981 \AddToHook{begindocument/before}{
982   \RenewDocumentEnvironment{quote}{!O{}}
983   { \UseInstance{blockenv}{quote} {#1} }
984   { \endblockenv }

985   \RenewDocumentEnvironment{quotation}{!O{}}
986   { \UseInstance{blockenv}{quotation} {#1} }
987   { \endblockenv }
988 }
```

### 9.4 Verbatim environments

```
verbatim (env.)
verbatim* (env.)
989 \AddToHook{begindocument/before}{
990   \RenewDocumentEnvironment{verbatim}{!O{}}
991   { \UseInstance{blockenv}{verbatim} {#1} }
992   { \endblockenv }

993   \RenewDocumentEnvironment{verbatim*}{!O{}}
994   { \UseInstance{blockenv}{verbatim*} {#1} }
995   { \endblockenv }
996 }
```

#### 9.4.1 Helper commands for verbatim

##### `\legacyverbatimsetup`

This code resembles the  $\text{\LaTeX}2_{\varepsilon}$  verbatim implementation with a slight twist: in  $\text{\LaTeX}2_{\varepsilon}$  each code line was a paragraph using `\leftskip=\@totalleftmargin`. This was possible because the whole environment was implemented as a trivlist. As this is no longer the case setting `\leftskip` would alter the layout of a surrounding list. So instead we need to make sure that the paragraph end is executed in a group so that any parshape setup is preserved.

```
997 <@@=
998 \def\legacyverbatimsetup{%
999   \language\l@nohyphenation
1000   \@tempwafalse
1001   \def\par{%
1002     \if@tempswa
1003       \leavevmode \null {\@@par}\penalty\interlinepenalty
1004     \else
1005       \@tempwattrue
1006       \ifhmode{\@@par}\penalty\interlinepenalty\fi
1007     \fi}%
1008   \let\do\@makeother \dospecials
1009   \obeylines \verbatim@font \@noligs
1010   \everypar \expandafter{\the\everypar \unpenalty}%
```

next should be a socket

```

1011 \frenchspacing
1012 \tl_set:Nn \l__tag_para_tag_tl {codeline}
1013 }
1014 <@@=block>

```

(End of definition for \legacyverbatimsetup. This function is documented on page 15.)

**\@setupverbinvisiblespace** In the pdfTeX engine we need to use \pdffakespace chars for the invisible spaces. In luatex we do not want this as it would lead to doubling the number of real space chars. In dvi-mode we do not want that either: with pdftex it would error, with xetex it does nothing.

```

1015 \newcommand\@setupverbinvisiblespace{}
1016 \bool_lazy_or:nnF
1017 { \sys_if_engine luatex_p: }
1018 { \sys_if_output_dvi_p: }
1019 {
1020   \renewcommand\@setupverbinvisiblespace
1021     {\def\@xobeysp{\nobreakspace\pdffakespace}}
1022 }

```

(End of definition for \@setupverbinvisiblespace. This function is documented on page 15.)

## 9.5 Standard list environments

**itemize** (*env.*)

**enumerate** (*env.*)

**description** (*env.*)

For the standard lists everything is managed by the blockenv instance.

```

1023 \AddToHook{begindocument/before}{
1024   \RenewDocumentEnvironment{itemize}{!0{}}
1025     { \UseInstance{blockenv}{itemize} {#1} }
1026     { \endblockenv }

1027   \RenewDocumentEnvironment{enumerate}{!0{}}
1028     { \UseInstance{blockenv}{enumerate} {#1} }
1029     { \endblockenv }

1030   \RenewDocumentEnvironment{description}{!0{}}
1031     { \UseInstance{blockenv}{description} {#1} }
1032     { \endblockenv }
1033 }

```

## 9.6 verse environment

**verse** (*env.*) The verse environment has not special tagging currently. It is defined as a simple standard list and takes the tagging from there. But it must be redefined so that \itemindent is correctly set.

```

1034 \AddToHook{begindocument/before}{
1035   \RenewDocumentEnvironment{verse}{!O{}}{
1036     {
1037       \let\\\@centercr
1038       \UseInstance{blockenv}{list}
1039       {
1040         item-indent=-1.5em,
1041         para-indent=-1.5em,
1042         item-vspace=Opt,
1043         right-margin = \leftmargin,
1044         left-margin  = \leftmargin+1.5em,
1045         #1
1046       }
1047       \item\relax
1048     }
1049   { \endblockenv }
1050 }

```

`list (env.)`

The legacy 2e list environment is more complicated as we have to get the extra arguments accounted for.

```

1051 \AddToHook{begindocument/before}{
1052   \RenewDocumentEnvironment{list}{O{ } m m }
1053   {

```

We do this by storing them away and then call the list instance. Inside this instance the `setup-code` key contains `\legacylistsetupcode`, which makes use of the stored values.

```

1054     \tl_set:Nn \l__block_legacy_env_params_tl
1055     {
1056       \tl_set:Nn \@itemlabel {#2}
1057       #3
1058     }
1059     \UseInstance{blockenv}{list} {#1}
1060   }
1061   { \endblockenv }
1062 }

```

`\legacylistsetupcode`

And here is the extra code for use in the list instance setup inside the key `setup-code`.

```

1063 \cs_new:Npn \legacylistsetupcode {

```

Reset values to defaults:

```

1064   \dim_zero:N \listparindent
1065   \dim_zero:N \rightmargin
1066   \dim_zero:N \itemindent

```

By default a `list` environment is not numbered, but this happens already in the block template.

```
1067 % \tl_set:Nn \@listctr {}
1068 % \legacy_if_set_false:n { @nmbrrlist } % needed if lists are nested
```

By default there is a simple definition for `\makelabel`. It can be overwritten in the second mandatory argument to the list environment (stored in `\l_block_legacy_env_params_tl`) and is used if the instance sets the compatibility key to true.

```
1069 \let\makelabel\@mklab % TODO: customize
```

Now we use the argument with parameter settings to update some or all of the above defaults:

```
1070 \l_block_legacy_env_params_tl
```

As we don't know much about this list we can only make a guess about the nature of the list and the setting of the tag name (default `list` rolemapped to `<L>`) and any tag attributes may have to be overwritten in the optional key/value argument. But we do have some hints to play with.

```
1071 \legacy_if:NTF { @nmbrrlist }
1072   { \tl_set:Nn \l__tag_L_attr_class_tl {enumerate} } % numbered list
1073   { \tl_if_empty:NTF \@itemlabel
1074     { \tl_set:Nn \l__tag_L_attr_class_tl {list} } % no label
1075     { \tl_set:Nn \l__tag_L_attr_class_tl {itemize} } % unnumbered,
1076                                           % unordered
1077   }
1078 }
```

*(End of definition for `\legacylistsetupcode`. This function is documented on page 15.)*

```
\l_block_legacy_env_params_tl
```

```
1079 \tl_new:N\l_block_legacy_env_params_tl
```

*(End of definition for `\l_block_legacy_env_params_tl`.)*

Replace with code not using `\list`

```
1080 \AddToHook{begindocument/before}{
1081   \RenewDocumentEnvironment{trivlist}{ !0{} }
1082     { \list[#1]{}
1083       {
1084         \dim_zero:N \leftmargin
1085         \dim_zero:N \labelwidth
1086         \cs_set_eq:NN \makelabel \use:n
1087       }
1088     }
1089   { \endblockenv }
1090 }
```



## 9.7 Theorem-like environments

Theorem-like environments are defined in L<sup>A</sup>T<sub>E</sub>X with the help of `\newtheorem` declarations. Internally they used a list with a single item. Using lists was convenient back then, but in a tagged document you end up with a strange structure. We therefore alter the mechanism.

`\newtheorem` This is a slightly streamlined version of `\newtheorem`, but it still uses a lot of the 2e code for now. Eventually this will change.

```

1091 \RenewDocumentCommand \newtheorem { m O{#1} m o }
1092 {
1093   \expandafter\@ifdefinable\csname #1\endcsname
1094   {
1095     \str_if_eq:nnTF{#1}{#2}
1096     {
1097       \@definecounter {#2}
1098       \IfNoValueTF {#4}
1099       { % @ynthm
1100         \tl_gset:ce { the #2 }
1101         {
1102           \@thmcounter{#2}
1103         }
1104       }
1105       { % @xnthm
1106         \@newctr{#1}[#4]
1107         \tl_gset:ce { the #2 }
1108         {
1109           \expandafter\noexpand\csname the#4\endcsname
1110           \@thmcountersep
1111           \@thmcounter{#2}
1112         }
1113       }
1114     }
1115     { % @othm
1116       \@ifundefined{c@#2}
1117       { \nocounterr{#2} }
1118       {
1119         \tl_gset:cn { the #1 }
1120         { \UseName { the #2 } }
1121       }
1122     }
1123     \global\@namedef{#1} { \@thm{#2}{#3} }
1124     \global\@namedef{end#1}{ \@endtheorem }
1125   }
1126 }

```

*(End of definition for `\newtheorem`.)*

`\@thm` `\@thm` executes `\refstepcounter` too early for hyperref and structure destinations: the generated target is outside the structure and can be separated from the theorem by a page break. We therefore move the anchor setting into `\@begintheorem`. `\@begintheorem` doesn't currently get the name of the counter as argument, so we store it in variable for now, to be able to pass it along.

```

1127 \tl_new:N \l__block_thm_current_counter_tl
1128 \def\@thm#1#2{%
1129   \@kernel@refstepcounter{#1}
1130   \tl_set:Nn \l__block_thm_current_counter_tl{#1}
1131   \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}]

```

To avoid that hyperref overwrites the definition again we must its patch:

```

1132 \def\hyper@nopatch@thm{}

(End of definition for \@thm.)

```

**\@begintheorem**  
**\@opargbegintheorem**

The \@thm command expands to either \@beginthorem or \@opargbegintheorem. For the moment we stick with this as it will help with the transition. But instead of using a `trivlist` we use a `blockenv` and some tagging for the title (as a `Caption`). We do not want potential tagging from `\textbf` here, so we use `\bfseries` to set the font. The commands set also the link targets which should be inside the main structure.

```

1133 \def\@begintheorem#1#2{
1134   \UseInstance{blockenv}{theorem}{}
1135   \tagpdfparaOff

1136   \noindent
1137   \MakeLinkTarget{\l__block_thm_current_counter_tl}
1138   \tag_struct_begin:n{tag=Caption}
1139   \group_begin:
1140   \bfseries
1141   \tag_mc_begin:n {}
1142   #1\
1143   \tag_mc_end:
1144   \tag_struct_begin:n{tag=Lbl}
1145   \tag_mc_begin:n {}
1146   #2
1147   \tag_mc_end:
1148   \tag_struct_end:
1149   \group_end:
1150   \tag_struct_end:
1151   \tagpdfparaOn

1152   \tag_socket_use:nn { block/startpara/direct } { \PARALABEL }

1153   \itshape
1154   \hskip\labelsep
1155   \ignorespaces
1156 }
1157 \def\@opargbegintheorem#1#2#3{
1158   \UseInstance{blockenv}{theorem}{}
1159   \tagpdfparaOff

```

```

1160 \noindent
1161 \MakeLinkTarget{\l_block_thm_current_counter_tl}
1162 \tag_struct_begin:n{tag=Caption}
1163 \group_begin:
1164 \bfseries
1165 \tag_mc_begin:n {}
1166 #1\
1167 \tag_mc_end:
1168 \tag_struct_begin:n{tag=Lbl}
1169 \tag_mc_begin:n {}
1170 #2
1171 \tag_mc_end:
1172 \tag_struct_end:
1173 \tag_mc_begin:n {}
1174 \ (#3)
1175 \tag_mc_end:
1176 \group_end:
1177 \tag_struct_end:
1178 \tagpdfpara0n

1179 \tag_socket_use:nn { block/startpara/direct} { \PARALABEL }

1180 \itshape
1181 \hskip\labelsep
1182 \ignorespaces
1183 }

1184 \def\@endtheorem{\endblockenv}

```

*(End of definition for \@begintheorem and \@opargbegintheorem.)*

## 10 Instance declarations for environments

### 10.1 Blockenv instances

The blockenv instances handle overall setup for the document-level environments, i.e.,

- name of the environment for debugging purposes (`name`)
- how tagging should be performed (`tagging-recipe`, `tag-name`, `tag-attr-class`)
- does this environment changes the block level if nested (`increment-level`)
- any special setup code; normally not used (`setup-code`)
- what kind of block instance should be used (`block-instance`)
- what kind of para instance should be used; empty means inherit from the outside (`para-instance`)
- are inner paragraphs real paragraphs (default) or are they just `<text>` structures and part of an outer paragraph (`tagging-suppress-paras`)
- what kind of inner instance should be used, if any (`inner-instance`, `inner-instance-type`)

Note that block does set some values — check if this is right!

- the counter name of the inner level, if any (`inner-level-counter`)
- supported nesting depth of the inner level, if relevant (`max-inner-levels`)
- extra code executed at the end, by default `\ignorespaces` (`final-code`)

The `blockenv displayblock` instance below shows the full set with those using default values being commented out.

### 10.1.1 Basic instances

`blockenv displayblock` (*inst.*) This is like  $\text{\LaTeX} 2_{\epsilon}$ 's `trivlist`, i.e., it produces a vertical block with default setting, but doesn't put a list inside but uses a `<Div>` structure.

```

1185 \DeclareInstance{blockenv}{displayblock}{display}
1186 {
1187   name                = displayblock,
1188   % tagging-recipe    = standard,
1189   % tag-name          = ,
1190   % tag-attr-class    = ,
1191   increment-level     = false,
1192   % setup-code        = ,
1193   % block-instance    = displayblock ,
1194   % para-instance     = ,
1195   % tagging-suppress-paras = false ,
1196   % inner-instance    = ,
1197   % inner-instance-type = list , % not relevant as there is no inner instance
1198   % inner-level-counter = ,      % not relevant as there is no inner instance
1199   % max-inner-levels   = 4,      % not relevant as there is no inner instance
1200   % final-code        = \ignorespaces ,
1201 }

```

`blockenv displayblockflattened` (*inst.*) This flattens inner paragraphs without any surrounding tag structure by using the basic tagging recipe.

```

1202 \DeclareInstance{blockenv}{displayblockflattened}{display}
1203 {
1204   name                = displayblockflattened,
1205   tag-name            = ,
1206   tag-attr-class      = ,
1207   tagging-recipe      = basic,
1208   tagging-suppress-paras = true ,
1209   inner-level-counter = ,
1210   increment-level     = false,
1211   setup-code          = ,
1212   block-instance      = displayblock ,
1213   inner-instance      = ,
1214 }

```

### 10.1.2 Center, flushleft, and flushright instances

All three environments use the `displayblock` instance as block instance. They only differ in the choice of para instance.

`blockenv center` (*inst.*) The `center` environment without using a list internally.

```

1215 \DeclareInstance{blockenv}{center}{display}
1216 {
1217   name                = center,
1218   tag-name            = ,
1219   tag-attr-class      = ,
1220   tagging-recipe      = basic,
1221   tagging-suppress-paras = true ,
1222   inner-level-counter = ,
1223   increment-level     = false,
1224   setup-code          = ,
1225   block-instance      = displayblock ,
1226   para-instance       = center ,
1227   inner-instance      = ,
1228 }

```

`blockenv flushleft` (*inst.*) The `flushleft` environment without using a list internally.

```

1229 \DeclareInstance{blockenv}{flushleft}{display}
1230 {
1231   name                = flushleft,
1232   tag-name            = ,
1233   tag-attr-class      = ,
1234   tagging-recipe      = basic,
1235   tagging-suppress-paras = true ,
1236   inner-level-counter = ,
1237   increment-level     = false,
1238   setup-code          = ,
1239   block-instance      = displayblock ,
1240   para-instance       = raggedright ,
1241   inner-instance      = ,
1242 }

```

`blockenv flushright` (*inst.*)

The `flushright` environment without using a list internally.

```

1243 \DeclareInstance{blockenv}{flushright}{display}
1244 {
1245   name                = flushleft,
1246   tag-name            = ,
1247   tag-attr-class      = ,
1248   tagging-recipe      = basic,
1249   tagging-suppress-paras = true ,
1250   inner-level-counter = ,
1251   increment-level     = false,
1252   setup-code          = ,
1253   block-instance      = displayblock ,
1254   para-instance       = raggedleft ,
1255   inner-instance      = ,
1256 }

```

### 10.1.3 Blockquote instances

$\LaTeX 2_{\epsilon}$  has two environments for quoting: `quote` and `quotation`. By default they differ only in indentation of inner paragraphs. This is handled by using separate block instances.

The tag names are both roll-mapped to `<BlockQuote>`.

`blockenv quotation` (*inst.*) For the quotation environment:

```
1257 \DeclareInstance{blockenv}{quotation}{display}
1258 {
1259   name           = quotation,
1260   tag-name       = quotation,
1261   tag-attr-class = ,
1262   tagging-recipe = standard,
1263   inner-level-counter = ,
1264   increment-level = true,
1265   setup-code     = ,
1266   block-instance = quotationblock ,
1267   inner-instance = ,
1268 }
```

`blockenv quote` (*inst.*) For the quote environment:

```
1269 \DeclareInstance{blockenv}{quote}{display}
1270 {
1271   name           = quote,
1272   tag-name       = quote,
1273   tag-attr-class = ,
1274   tagging-recipe = standard,
1275   inner-level-counter = ,
1276   increment-level = true,
1277   setup-code     = ,
1278   block-instance = quoteblock ,
1279   inner-instance = ,
1280 }
```

#### 10.1.4 The theorem instance

`blockenv theorem` (*inst.*)

We use `<theorem-like>` as the structure name and rolemap it to a `<Sect>` because that can hold a `<Caption>`.

```
1281 \DeclareInstance{blockenv}{theorem}{display}
1282 {
1283   name           = theorem-like,
1284   tag-name       = theorem-like,
1285   tag-attr-class = ,
1286   tagging-recipe = standalone,
1287   inner-level-counter = ,
1288   increment-level = false,
1289   setup-code     = ,
1290   block-instance = theoremblock ,
1291 }
```

### 10.1.5 The `verbatim` and `verbatim*` instances

The rolemapping is `<verbatim>` to `<Code>` and `<codeline>` to `<Sub>` (which is role mapped to `<Span>` in pdf 1.7). Sub inside Code is allowed according the errata of ISO 32005. The paragraphs inside `verbatim` are flattened. Line numbers should be inside the `<codeline>` structure and be tagged either as `<Lb1>` or `<Artifact><Lb1>`.

`blockenv verbatim` (*inst.*)

```
1292 \DeclareInstance{blockenv}{verbatim}{display}
1293 {
1294   name                = verbatim,
1295   tag-name            = verbatim,
1296   tag-attr-class     = ,
1297   tagging-recipe     = standard,
1298   tagging-suppress-paras = true,
1299   inner-level-counter = ,
1300   increment-level    = false,
1301   setup-code        = ,
1302   block-instance    = verbatimblock ,
1303   inner-instance    = ,
1304   para-instance     = justify ,
1305   final-code       = \legacyverbatimsetup
```

Here is where `verbatim` and `verbatim*` technically differ: in the former we set up spaces to become nobreakable spaces (if necessary followed by a `\pdfmakespace` in the pdf $\TeX$  engine) and in `verbatim*` we set it up to generate visible space chars.

```
1306                                     \@setupverbinvisiblespace \@vobeyspaces
```

Then we start the special scanning process to look for `\end{verbatim}` with special catcodes and grab everything inbetween. For `verbatim*` we use `\@sxverbatim` to look for `\end{verbatim*}` instead.<sup>1</sup>

```
1307                                     \@sxverbatim
1308 }
```

`blockenv verbatim*` (*inst.*)

```
1309 \DeclareInstance{blockenv}{verbatim*}{display}
1310 {
1311   name                = verbatim,
1312   tag-name            = verbatim,
1313   tag-attr-class     = ,
1314   tagging-recipe     = standard,
1315   tagging-suppress-paras = true,
1316   inner-level-counter = ,
1317   increment-level    = false,
1318   setup-code        = ,
1319   block-instance    = verbatimblock ,
1320   inner-instance    = ,
```

---

<sup>1</sup>Perhaps there should be some other command names for this?

```

1321 para-instance      = justify ,
1322 final-code         = \legacyverbatimsetup
1323                   \@setupverbvisiblespace \@vobeyspaces
1324                   \@sxverbatim
1325 }

```

### 10.1.6 Standard list instances

`blockenv itemize` (*inst.*) For the `itemize` environment:

```

1326 \DeclareInstance{blockenv}{itemize}{display}
1327 {
1328   name           = itemize,
1329   tag-name       = itemize,
1330   tag-attr-class = itemize,
1331   tagging-recipe = list,
1332   inner-level-counter = \@itemdepth,
1333   increment-level = true,
1334   max-inner-levels = 4,
1335   setup-code     = ,
1336   block-instance = listblock ,
1337   inner-instance = itemize ,
1338 }

```

`blockenv enumerate` (*inst.*)

For the `enumerate` environment:

```

1339 \DeclareInstance{blockenv}{enumerate}{display}
1340 {
1341   name           = enumerate,
1342   tag-name       = enumerate,
1343   tag-attr-class = enumerate,
1344   tagging-recipe = list,
1345   increment-level = true,
1346   max-inner-levels = 4,
1347   setup-code     = ,
1348   block-instance = listblock ,
1349   inner-level-counter = \@enumdepth,
1350   inner-instance = enum ,
1351 }

```

`blockenv description` (*inst.*)

For the `description` environment:

```

1352 \DeclareInstance{blockenv}{description}{display}
1353 {
1354   name           = description,
1355   tag-name       = description,
1356   tag-attr-class = description,
1357   tagging-recipe = list,
1358   inner-level-counter = ,
1359   increment-level = true,
1360   setup-code     = ,

```



```

1361 block-instance      = listblock ,
1362 inner-instance      = description ,
1363 }

```

`blockenv list` (*inst.*) The general (legacy) `list` environment does some of its setup in the `setup-code` key.

```

1364 \DeclareInstance{blockenv}{list}{display}
1365 {
1366   name              = list,
1367   tag-name          = list,
1368   tag-attr-class    = ,
1369   tagging-recipe    = list,
1370   increment-level   = true,
1371   setup-code        = \legacylistsetupcode ,
1372   block-instance    = listblock ,
1373   inner-level-counter = ,
1374   inner-instance    = legacy ,
1375 }

```

## 10.2 Block instances

Below, we declare the different block instances for the document-level environments. Some, such as the `displayblock` ones are shared between different environments (as long as one doesn't need to adjust individual values), other environments have their own instances (for precisely that reason).

### 10.2.1 Displayblock instances

We provide 6 nesting levels (as in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>). If you want to provide more you need to change the `maxblocklevels` counter, offer further `displayblock-xx` instances but also define further (legacy) `\list<romannumeral>` commands for the defaults. If not, then the settings from the previous level are reused automatically—which may or may not be good enough).

```

1376 \setcounter{maxblocklevels}{6}

```

`block displayblock-0` (*inst.*)

`block displayblock-1` (*inst.*)

Here we need level zero as well in case a flattened `displayblock` (like the `center` env) it is used on top-level.

`block displayblock-2` (*inst.*)

`block displayblock-3` (*inst.*)

We show all keys here for reference, with those using their default values commented out:

`block displayblock-4` (*inst.*)

`block displayblock-5` (*inst.*)

`block displayblock-6` (*inst.*)

```

1377 \DeclareInstance{block}{displayblock-0}{display}
1378 {
1379 %   begin-vspace      = \topsep ,
1380 %   begin-extra-vspace = \partopsep ,
1381 %   para-vspace       = \parsep ,
1382 %   end-vspace        = \KeyValue{begin-vspace} ,
1383 %   end-extra-vspace  = \KeyValue{begin-extra-vspace} ,
1384 %   item-vspace       = \itemsep ,
1385 %   begin-penalty     = \UserName{@beginparpenalty} ,
1386 %   end-penalty       = \UserName{@endparpenalty} ,
1387   left-margin        = Opt ,

```

```

1388 %   right-margin      = \rightmargin ,
1389 %   para-indent      = Opt ,
1390 }

1391 \DeclareInstanceCopy{block}{displayblock-1}{displayblock-0}
1392 \DeclareInstanceCopy{block}{displayblock-2}{displayblock-0}
1393 \DeclareInstanceCopy{block}{displayblock-3}{displayblock-0}
1394 \DeclareInstanceCopy{block}{displayblock-4}{displayblock-0}
1395 \DeclareInstanceCopy{block}{displayblock-5}{displayblock-0}
1396 \DeclareInstanceCopy{block}{displayblock-6}{displayblock-0}

```

### 10.2.2 Verbatim instances

Verbatim instances have their own levels so that one can specify specific indentations or vertical separations between lines.

```

block verbatimblock-0 (inst.)
block verbatimblock-1 (inst.)
block verbatimblock-2 (inst.) \DeclareInstance{block}{verbatimblock-0}{display}
block verbatimblock-3 (inst.) {
block verbatimblock-4 (inst.)   left-margin      = Opt ,
block verbatimblock-5 (inst.)   para-vspace      = Opt ,
block verbatimblock-6 (inst.) }

1402 \DeclareInstanceCopy{block}{verbatimblock-1}{verbatimblock-0}
1403 \DeclareInstanceCopy{block}{verbatimblock-2}{verbatimblock-0}
1404 \DeclareInstanceCopy{block}{verbatimblock-3}{verbatimblock-0}
1405 \DeclareInstanceCopy{block}{verbatimblock-4}{verbatimblock-0}
1406 \DeclareInstanceCopy{block}{verbatimblock-5}{verbatimblock-0}
1407 \DeclareInstanceCopy{block}{verbatimblock-6}{verbatimblock-0}

```

### 10.2.3 Quote/quotationblock instances

Quote and quotation are not flattened, i.e., they change levels, thus they start with level 1 not 0.

block quoteblock-1 (inst.) Default layout is to indent equally from both sides.

```

block quoteblock-2 (inst.)
block quoteblock-3 (inst.) \DeclareInstance{block}{quoteblock-1}{display}
block quoteblock-4 (inst.) { right-margin = \KeyValue{left-margin} }
block quoteblock-5 (inst.) \DeclareInstanceCopy{block}{quoteblock-2}{quoteblock-1}
block quoteblock-6 (inst.) \DeclareInstanceCopy{block}{quoteblock-3}{quoteblock-1}
1412 \DeclareInstanceCopy{block}{quoteblock-4}{quoteblock-1}
1413 \DeclareInstanceCopy{block}{quoteblock-5}{quoteblock-1}
1414 \DeclareInstanceCopy{block}{quoteblock-6}{quoteblock-1}

```

block quotationblock-1 (inst.) Quotation additionally changes the para-indent.

```

block quotationblock-2 (inst.)
block quotationblock-3 (inst.) \DeclareInstance{block}{quotationblock-1}{display}
block quotationblock-4 (inst.) { para-indent = 1.5em , right-margin = \KeyValue{left-margin} }
block quotationblock-5 (inst.) \DeclareInstanceCopy{block}{quotationblock-2}{quotationblock-1}
block quotationblock-6 (inst.) \DeclareInstanceCopy{block}{quotationblock-3}{quotationblock-1}
1419 \DeclareInstanceCopy{block}{quotationblock-4}{quotationblock-1}
1420 \DeclareInstanceCopy{block}{quotationblock-5}{quotationblock-1}
1421 \DeclareInstanceCopy{block}{quotationblock-6}{quotationblock-1}

```

## 10.2.4 Block instances for the theorems

`block theoremblock-0` (*inst.*) Theorems do not support nesting, so we have only one to set up. The L<sup>A</sup>T<sub>E</sub>X default reused the general value of `\parindent` and `\parskip` and, of course, they start at the outer margin.

```
1422 \DeclareInstance{block}{theoremblock-0}{display}
1423 {
1424     left-margin      = Opt ,
1425     para-indent      = \parindent ,
1426     para-vspace      = \parskip ,
1427 }
```

There are, however, documents that put theorem-like environments inside of lists. While that is in most case somewhat dubious, it can make sense, for example, in `description` lists. So we support it somewhat by also providing `theoremblock` instances for level 1 and 2. If somebody really nests them further down, then more such instances need to be declared.

```
1428 \DeclareInstanceCopy{block}{theoremblock-1}{theoremblock-0}
1429 \DeclareInstanceCopy{block}{theoremblock-2}{theoremblock-0}
```

## 10.2.5 Block instances for the standard lists

`block listblock-1` (*inst.*) The block instances for the various list environments use the same underlying instance `block listblock-2` (*inst.*) (well, by default) and nothing needs to be set up specifically (because that is already done in the legacy `\list{romannumeral}` unless a different layout is wanted.

`block listblock-4` (*inst.*)

```
block listblock-5 (inst.) \DeclareInstance{block}{listblock-1}{display}{
block listblock-6 (inst.) % begin-vspace      = \topsep ,
1432 % begin-extra-vspace = \partopsep ,
1433 % para-vspace        = \parsep ,
1434 % end-vspace         = \KeyValue{begin-vspace} ,
1435 % end-extra-vspace   = \KeyValue{begin-extra-vspace} ,
1436 % item-vspace        = \itemsep ,
1437 % begin-penalty      = \UseName{@beginparpenalty} ,
1438 % end-penalty        = \UseName{@endparpenalty} ,
1439 % left-margin        = \leftmargin ,
1440 % right-margin       = \rightmargin ,
1441 % para-indent        = Opt ,
1442 }
1443 \DeclareInstance{block}{listblock-2}{display}{}
1444 \DeclareInstance{block}{listblock-3}{display}{}
1445 \DeclareInstance{block}{listblock-4}{display}{}
1446 \DeclareInstance{block}{listblock-5}{display}{}
1447 \DeclareInstance{block}{listblock-6}{display}{}

```

## 10.3 List instances for the standard lists

For all list instances we have to say what kind of label we want (`item-label`) and how it should be formatted.

`list itemize-1` (*inst.*) For `itemize` environments this is all we need to do and we refer back to the external `list itemize-2` (*inst.*) definitions rather than defining the `item-label` code in the instance to ensure that old `list itemize-3` (*inst.*) documents still work.

`list itemize-4` (*inst.*)

```
1448 \DeclareInstance{list}{itemize-1}{std}{ item-label = \labelitemi }
1449 \DeclareInstance{list}{itemize-2}{std}{ item-label = \labelitemii }
1450 \DeclareInstance{list}{itemize-3}{std}{ item-label = \labelitemiii }
1451 \DeclareInstance{list}{itemize-4}{std}{ item-label = \labelitemiv }
```

`list enumerate-1` (*inst.*) `enumerate` environments are similar, except that we also have to say which counter to use on each level.

`list enumerate-2` (*inst.*)

`list enumerate-3` (*inst.*)

```
list enumerate-4 (inst.) \DeclareInstance{list}{enum-1}{std}
1453   { item-label = \labelenumi , counter = enumi }
1454 \DeclareInstance{list}{enum-2}{std}
1455   { item-label = \labelenumii , counter = enumii }
1456 \DeclareInstance{list}{enum-3}{std}
1457   { item-label = \labelenumiii , counter = enumiii }
1458 \DeclareInstance{list}{enum-4}{std}
1459   { item-label = \labelenumiv , counter = enumiv }
```

`list legacy` (*inst.*) For the legacy `list` environment there is only one instance which is reused on all levels. This is done this way because the legacy `list` environment sets all its parameters through its arguments. So this instances shouldn't really be touched. It sets the `legacy-support` key to true, which means that the list code uses `\makeLabel` for formatting the label.

```
1460 \DeclareInstance{list}{legacy}{std} {
1461   item-instance = basic ,
1462   legacy-support = true ,
1463 }
```

`list description` (*inst.*) The `description` lists also use only a single list instance with only one key not using the default:

```
1464 \DeclareInstance{list}{description}{std} { item-instance = description }
```

## 10.4 Item instances

`item basic` (*inst.*) There two item instances set up: `description` for use with the `description` environment and `basic` for use with all other lists (up to now).

```
1465 \DeclareInstance{item}{basic}{std}
1466   { label-align = right }

1467 \DeclareInstance{item}{description}{std}
1468   {
1469     label-format = \normalfont\bfseries #1 ,
1470     label-align = left
1471   }
```

## 10.5 Para instances

```
1472 \tagpdfsetup
1473 {
1474   role/new-attribute = {justify}    {/0 /Layout /TextAlign/Justify},
1475   role/new-attribute = {center}     {/0 /Layout /TextAlign/Center},
1476   role/new-attribute = {raggedright}{/0 /Layout /TextAlign/Start},
1477   role/new-attribute = {raggedleft}{/0 /Layout /TextAlign/End},
1478 }
```

para center (*inst.*)

```
1479 \DeclareInstance{para}{center}{std}
1480 {
1481   para-attr-class      = center ,
1482   para-indent          = Opt ,
1483   % begin-hspace       = Opt ,
1484   left-hspace          = \@flushglue ,
1485   right-hspace         = \@flushglue ,
1486   end-hspace           = \z@skip ,
1487   final-hyphen-demerits = 0 ,
1488   newline-cmd          = \@centercr ,
1489 }
```

para raggedright (*inst.*)

```
1490 \DeclareInstance{para}{raggedright}{std}
1491 {
1492   para-attr-class      = raggedright ,
1493   para-indent          = Opt ,
1494   % begin-hspace       = Opt ,
1495   left-hspace          = \z@skip ,
1496   right-hspace         = \@flushglue ,
1497   end-hspace           = \z@skip ,
1498   final-hyphen-demerits = 0 ,
1499   newline-cmd          = \@centercr ,
1500 }
```

para raggedleft (*inst.*)

```
1501 \DeclareInstance{para}{raggedleft}{std}
1502 {
1503   para-attr-class      = raggedleft ,
1504   para-indent          = Opt ,
1505   % begin-hspace       = Opt ,
1506   left-hspace          = \@flushglue ,
1507   right-hspace         = \z@skip ,
1508   end-hspace           = \z@skip ,
1509   final-hyphen-demerits = 0 ,
1510   newline-cmd          = \@centercr ,
1511 }
```

para justify (*inst.*) Justifying is exactly what the default values do, so the instance hasn't any special setup.

```
1512 \DeclareInstance{para}{justify}{std}
1513 {
1514   % para-attr-class      = justify ,
```

```

1515 % para-indent          = \parindent ,
1516 % begin-hspace        = Opt ,
1517 % left-hspace         = \z@skip ,
1518 % right-hspace        = \z@skip ,
1519 % end-hspace          = \flushglue ,
1520 % final-hyphen-demerits = 5000 ,
1521 % newline-cmd         = \@normalcr ,
1522 }

\centering
\raggedleft1523 \DeclareRobustCommand\centering {\UseInstance{para}{center}{}}
\raggedright1524 \DeclareRobustCommand\raggedleft {\UseInstance{para}{raggedleft}{}}
\justifying1525 \DeclareRobustCommand\raggedright{\UseInstance{para}{raggedright}{}}
1526 \DeclareRobustCommand\justifying {\UseInstance{para}{justify}{}}

1527 \justifying

      (End of definition for \centering and others.)

1528 </package>

1529 <*latex-lab>
1530 \ProvidesFile{block-latex-lab-testphase.ltx}
1531     [\ltxlabblockdate\space v\ltxlabblockversion\space
1532         blockenv implementation]
1533 \RequirePackage{latex-lab-testphase-block}
1534 </latex-lab>

```

## A Documentation from first prototype implementations

### A.1 Open questions

- Existing questions — moved to issues —

### A.2 Code cleanup

- Actually implement what’s announced.
- Encapsulate most uses of `\legacy_if...` into commands with `expl3` syntax: we cannot rename these booleans for compatibility reasons but we can make the code cleaner nevertheless. — made issue —
- The `\topsep` and `\partopsep` business is tricky to reproduce exactly (see `\@topsepadd` and `\@topsep`) because of how it accumulates when lists are nested immediately.

### A.3 Tasks

- Change author to LaTeX Team once it’s nice enough to deserve that label.
- Reproducing exactly the standard layouts and examples in the `enumitem` documentation.

- Hooks, but do not duplicate those that already exist as environment hooks. Hence, mostly around items.
- Customization and interaction with LDB:
  - Allow arbitrary nesting depth with automatically defined styles for labels, counters etc.
  - Adapt everything to font size! (e.g. footnotes).
  - How to model the inheritance from `trivlist` to `list` to `enumerate`?
- Add key–value settings mimicking `enumitem`'s ability to set any four of five horizontal parameters and deduce the fifth by  $\text{\leftmargin} + \text{\itemindent} = \text{\labelindent} + \text{\labelwidth} + \text{\labelsep}$ .
- Provide good ways to customize how overlong labels are dealt with.
- Use the `.aux` file.
  - Implement the `\ref` styles that `enumitem` provides.
  - Reverse enumerations, important in publication lists and the like. Somehow avoid needing 3 compilations for references to reverse enumerations to settle?
  - Ability to calculate `\labelwidth` from the label contents. Share calculated parameters between multiple environments (cf. `resume` option).
- Related to grabbing the whole list environment, and input syntax variations:
  - Other layouts: `tabular` (see `listliketab` vs `typed-checklist`), `multicolumn` and `horizontally numbered` (see `tasks`), `inline lists`, `runin lists` in the easy case where there is no intervening `\par`.
  - Formatting the item text in a box or similar (requires grabbing the whole list).
  - Filtering which items to show: hide certain items according to criteria (useful together with `list reuse`), see `typed-checklist`.
  - Shorthands `\iitem` for automatic nested lists, or `\1`, `\2` etc from `outlines`.
  - Support markdown input like `asciilist`.
- Check interaction with `babel` options such as `french` or `accadian` (see `FrenchItemizeSpacing`)
- RTL and vertical typesetting.

## B Plan of attack of first prototype

Typesetting list environments involves a rather large number of parameters. They can be affected by the context such as the total list nesting level, the nesting level of the given type of list, and the font size. An environment like `enumerate` has two main aspects.

- It has a certain layout in the page, with vertical and horizontal spacing around it. This type of layout is shared with environments such as `quote`, `flushright`, or `tabbing`. This common layout is implemented in  $\text{\LaTeX} 2_{\epsilon}$  through `\trivlist` (or `\list`).

- It defines how each `\item` should be typeset: how to construct the label, in particular the `counter` name, and how to format the content of the item.

This suggests defining two template types, *block* and *item* covering these two aspects.<sup>2</sup> While the *item* type will perhaps have a single template, one could typeset a *block* template in several ways, for instance the standard L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> way or a fancy colored box.

The *general block* template should receive the following parameters. The *plain block* template is a restricted template that freezes all item-related parameters to dummy values (`counter`, `start`, `resume`, `label-width`, `label-sep` and all `item-*`). The *list block* template is a restricted template<sup>3</sup> that omits the `heading` parameter and whose default for `item-instance` is non-empty.

- Structural parameters: the `heading` to place before, `counter` name, `start` value, whether to `resume` a previous list, and the `item-instance` (an *item* instance) to use when typesetting items.
- Vertical spacing and penalties: `begin-penalty`, `begin-vspace`, `begin-extra-vspace`, `item-penalty`, `item-vspace`, `item-par-skip`, `end-penalty`, `end-vspace`, `end-extra-vspace`.
- Horizontal spacing: `right-margin`, `left-margin`, `para-indent`, `item-indent`, `label-width`, `label-sep`.

document class customizations

A document class should edit these templates (or define restricted templates) to set up default values that depend on `\g_block_nesting_depth_int`, namely how many lists are nested overall.<sup>4</sup> The document class should then set up an instance of these templates for each environment, with appropriate settings such as a `heading`, a suitable `item-instance`, or making `margin-right` equal to `margin-left` in a quote environment.

The *inline-list block* template receives many fewer parameters. Note that `begin-vspace`, `item-vspace`, `end-vspace` are now *horizontal skips*.

- Structural parameters: `counter`, `start`, `resume`, `item-instance`.
- Spacing and penalties: `begin-penalty`, `begin-vspace`, `item-penalty`, `item-vspace`, `end-penalty`, `end-vspace`.
- Horizontal spacing: `label-width`, `label-sep`.

The *std item* template should receive the following parameters. They depend on the type of list and its nesting level among lists of such type, but typically not on the total nesting level.

- Counter name (`counter`), shared with the parent *list block* template, but needed for incrementing.
- Label construction: a function `counter-label` that produces the label from the counter name, used if `\item` is given without argument.

<sup>2</sup>Possibly also *endblock* to deal with decorations at the end?

<sup>3</sup>A better approach could be to have a notion of inheritance for template types, so that we end up with two different *template types*. Then we can implement other template for the list template type: *table* for lists typeset as rows/columns of a table, *inline* for lists typeset in horizontal mode within a paragraph, and *runin* for run-in lists.

<sup>4</sup>Does `xtemplate` provide a way to specify default values that are only evaluated once an instance is used?

Text wrong and or concept with vspace and hspace questionable!

revisit!

check!



- References: a function `counter-ref` for how the label should be referred to when it is constructed from the counter, `label-ref` and `label-autoref` used when `\item` has an optional argument.
- Label formatting: `label-format` function, `label-strut` boolean.
- Label alignment (`label-align`, `label-boxed`, `next-line`).
- Content parameters: `text-font`.
- A `compatibility` boolean that controls for instance whether `\makelabel` is used.

document class  
customizations

The document class should set up an instance such as *enumiii* for each environment and nesting level.<sup>5</sup>

A given environment will adjust some nesting levels, then call the *block* instance appropriate to the environment type, passing it the *item* instance appropriate to the environment and depth. Additional context-dependence could be provided by `l3ldb`, but the main context-dependence should not rely on it for simplicity reasons and incidentally because `l3ldb` is not yet available.

## Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

	<b>Symbols</b>				
<code>\</code>	.....	308, 497, 498, 1037		<code>\g_block_nesting_depth_int</code>	.....
<code>\u</code>	.....	743, 1142, 1166, 1174		.....	15, <u>167</u> , 209, 213, 215, 225, 260
			<b>Numbers</b>	<code>block display (template)</code>	.....
<code>\1</code>	.....	71		<code>block displayblock-0 (instance)</code>	... <u>1377</u>
<code>\2</code>	.....	71		<code>block displayblock-1 (instance)</code>	... <u>1377</u>
			<b>A</b>	<code>block displayblock-2 (instance)</code>	... <u>1377</u>
<code>\addpenalty</code>	.....	282, 399, 711		<code>block displayblock-3 (instance)</code>	... <u>1377</u>
<code>\AddToHook</code>	.....	649, 680, 776, 778, 970, 981, 989, 1023, 1034, 1051, 1080		<code>block displayblock-4 (instance)</code>	... <u>1377</u>
<code>\addvspace</code>	.....	283, 397, 400, 401, 712		<code>block displayblock-5 (instance)</code>	... <u>1377</u>
<code>\arabic</code>	.....	127		<code>block displayblock-6 (instance)</code>	... <u>1377</u>
<code>\AssignTaggingSocketPlug</code>	.....	.....		<code>block internal commands:</code>	
.....	.....	57, 647, 775, 798, 910		<code>\__block_beginpar_hmode:N</code>	.....
			<b>B</b>	.....	736, 736, 913, 935, 949
<code>\begin</code>	.....	15, 45, <u>825</u>		<code>\__block_beginpar_vmode:</code>	.....
<code>\begingroup</code>	.....	834		.....	716, 716, 915, 937, 951
<code>\bfseries</code>	.....	1140, 1164, 1469		<code>\l__block_block_instance_tl</code>	.....
<code>block (templatetype)</code>	.....	<u>61</u>		.....	177, 224, 289
<code>block commands:</code>				<code>\l__block_botsep_skip</code>	.....
<code>\block_debug_off:</code>	..	15, <u>147</u> , 152, 165		.....	320, 508
<code>\block_debug_on:</code>	...	15, <u>147</u> , 147, 164		<code>\__block_counter_label:n</code>	..
				.....	516, 553
				<code>\__block_counter_ref:n</code>	.....
				.....	517
				<code>\l__block_counter_start_int</code>	.....
				.....	436, 470, 473, 482
				<code>\l__block_counter_tl</code>	...
				.....	434, 468, 478
				<code>\__block_debug:n</code>	.....
				.....	<u>145</u> , 145, 159

<sup>5</sup>This should be made easily extendible to deeper levels.

<code>\g_block_debug_bool</code> .....	<code>\l_block_level_incr_bool</code> .....
..... <a href="#">144</a> , <a href="#">149</a> , <a href="#">154</a> , <a href="#">160</a> , <a href="#">162</a>	..... <a href="#">175</a> , <a href="#">207</a> , <a href="#">259</a>
<code>\__block_debug_gset</code> : <a href="#">147</a> , <a href="#">150</a> , <a href="#">155</a> , <a href="#">157</a>	<code>\__block_list_begin</code> : ... <a href="#">872</a> , <a href="#">872</a> , <a href="#">952</a>
<code>\__block_debug_typeout:n</code> . <a href="#">41</a> , <a href="#">48</a> ,	<code>\__block_list_end</code> : ... <a href="#">889</a> , <a href="#">889</a> , <a href="#">953</a>
<a href="#">145</a> , <a href="#">146</a> , <a href="#">161</a> , <a href="#">187</a> , <a href="#">258</a> , <a href="#">272</a> , <a href="#">291</a> ,	<code>\__block_list_item_begin</code> : .....
<a href="#">412</a> , <a href="#">416</a> , <a href="#">420</a> , <a href="#">448</a> , <a href="#">491</a> , <a href="#">534</a> , <a href="#">556</a> ,	..... <a href="#">879</a> , <a href="#">879</a> , <a href="#">954</a>
<a href="#">652</a> , <a href="#">717</a> , <a href="#">740</a> , <a href="#">766</a> , <a href="#">847</a> , <a href="#">851</a> , <a href="#">885</a> , <a href="#">896</a>	<code>\__block_list_item_end</code> : <a href="#">880</a> , <a href="#">880</a> , <a href="#">955</a>
<code>\l_block_effective_top_skip</code> ...	<code>\l_block_long_label_bool</code> .....
..... <a href="#">350</a> , <a href="#">352</a> , <a href="#">400</a> , <a href="#">678</a>	..... <a href="#">582</a> , <a href="#">583</a> , <a href="#">591</a> , <a href="#">603</a>
<code>\l_block_env_name_tl</code> .....	<code>\__block_make_label_box:n</code> .....
..... <a href="#">171</a> , <a href="#">187</a> , <a href="#">462</a> , <a href="#">557</a>	..... <a href="#">36</a> , <a href="#">550</a> , <a href="#">552</a> , <a href="#">557</a> , <a href="#">604</a> , <a href="#">604</a>
<code>\__block_evaluate_saved_user_</code> -	<code>\l_block_max_inner_levels_tl</code> ..
<code>keys:nn</code> .....	..... <a href="#">181</a> , <a href="#">203</a>
<a href="#">32</a> , <a href="#">33</a> ,	<code>\l_block_next_line_bool</code> .. <a href="#">523</a> , <a href="#">590</a>
<a href="#">36</a> , <a href="#">425</a> , <a href="#">425</a> , <a href="#">428</a> , <a href="#">429</a> , <a href="#">451</a> , <a href="#">454</a> , <a href="#">536</a>	<code>\l_block_one_label_box</code> .....
<code>\l_block_final_code_tl</code> . <a href="#">25</a> , <a href="#">184</a> , <a href="#">248</a>	..... <a href="#">38</a> , <a href="#">563</a> , <a href="#">567</a> , <a href="#">569</a> ,
<code>\__block_if_list:TF</code> <a href="#">266</a> , <a href="#">271</a> , <a href="#">288</a> , <a href="#">288</a>	<a href="#">573</a> , <a href="#">574</a> , <a href="#">578</a> , <a href="#">579</a> , <a href="#">581</a> , <a href="#">588</a> , <a href="#">601</a> , <a href="#">606</a>
<code>\__block_inner_begin</code> : .....	<code>\l_block_para_instance_tl</code> ....
..... <a href="#">843</a> , <a href="#">843</a> , <a href="#">925</a> , <a href="#">938</a>	..... <a href="#">178</a> , <a href="#">228</a> , <a href="#">230</a>
<code>\__block_inner_end</code> : <a href="#">846</a> , <a href="#">846</a> , <a href="#">926</a> , <a href="#">939</a>	<code>\l_block_parbotsep_skip</code> .. <a href="#">321</a> , <a href="#">509</a>
<code>\l_block_inner_instance_tl</code> ....	<code>\l_block_parindent_dim</code> ... <a href="#">327</a> , <a href="#">373</a>
..... <a href="#">183</a> , <a href="#">232</a> , <a href="#">236</a>	<code>\__block_recipe_basic</code> : .... <a href="#">911</a> , <a href="#">911</a>
<code>\l_block_inner_instance_type_tl</code>	<code>\__block_recipe_list</code> : .... <a href="#">946</a> , <a href="#">946</a>
..... <a href="#">182</a> , <a href="#">235</a>	<code>\__block_recipe_standalone</code> : <a href="#">920</a> , <a href="#">920</a>
<code>\l_block_inner_level_counter_tl</code>	<code>\__block_recipe_standard</code> : . <a href="#">932</a> , <a href="#">932</a>
..... <a href="#">180</a> , <a href="#">200</a> , <a href="#">202</a> , <a href="#">205</a> , <a href="#">237</a> , <a href="#">239</a>	<code>\l_block_resume_bool</code> .....
<code>\__block_inter_item</code> : <a href="#">41</a> , <a href="#">695</a> , <a href="#">704</a> , <a href="#">704</a>	..... <a href="#">437</a> , <a href="#">479</a>
<code>\l_block_item_align_tl</code> .....	<code>\__block_save_user_keys:n</code> .....
..... <a href="#">527</a> , <a href="#">528</a> , <a href="#">529</a> , <a href="#">571</a> , <a href="#">575</a> , <a href="#">600</a>	..... <a href="#">426</a>
<code>\l_block_item_compatibility_</code> -	<code>\l_block_setup_code_tl</code> . <a href="#">23</a> , <a href="#">176</a> , <a href="#">222</a>
<code>bool</code> .....	<code>\__block_skip_remove_last</code> : ....
..... <a href="#">525</a> , <a href="#">547</a>	..... <a href="#">139</a> , <a href="#">142</a> , <a href="#">268</a> , <a href="#">339</a> , <a href="#">707</a> , <a href="#">708</a>
<code>\__block_item_everypar</code> : .... <a href="#">34</a> ,	<code>\__block_skip_set_to_last:N</code> ....
<a href="#">38</a> , <a href="#">40</a> , <a href="#">41</a> , <a href="#">490</a> , <a href="#">598</a> , <a href="#">648</a> , <a href="#">648</a> , <a href="#">649</a> , <a href="#">671</a>	..... <a href="#">139</a> , <a href="#">139</a> , <a href="#">276</a> , <a href="#">389</a>
<code>\__block_item_everypar_first</code> : ..	<code>\l_block_tag_class_tl</code> . <a href="#">173</a> , <a href="#">960</a> , <a href="#">962</a>
..... <a href="#">490</a> , <a href="#">648</a> , <a href="#">674</a>	<code>\l_block_tag_inner_tag_tl</code> .....
<code>\__block_item_everypar_std</code> : ....	..... <a href="#">844</a> , <a href="#">929</a> , <a href="#">930</a> , <a href="#">942</a> , <a href="#">943</a> , <a href="#">945</a>
..... <a href="#">598</a> , <a href="#">648</a> , <a href="#">651</a>	<code>\l_block_tag_name_tl</code> .....
<code>\__block_item_instance:n</code> .....	..... <a href="#">172</a> , <a href="#">928</a> , <a href="#">930</a> , <a href="#">941</a> , <a href="#">943</a> , <a href="#">957</a> , <a href="#">959</a>
..... <a href="#">41</a> , <a href="#">438</a> , <a href="#">684</a> , <a href="#">697</a> , <a href="#">698</a>	<code>\l_block_tagging_recipe_tl</code> <a href="#">174</a> , <a href="#">218</a>
<code>\l_block_item_label_tl</code> <a href="#">435</a> , <a href="#">485</a> , <a href="#">487</a>	<code>\l_block_text_font_tl</code> .....
<code>\l_block_item_parsep_skip</code> .... <a href="#">595</a>	..... <a href="#">524</a>
<code>\__block_label_autoref:n</code> .....	<code>\l_block_thm_current_counter_tl</code>
..... <a href="#">519</a>	..... <a href="#">1127</a> , <a href="#">1130</a> , <a href="#">1137</a> , <a href="#">1161</a>
<code>\l_block_label_boxed_bool</code> <a href="#">522</a> , <a href="#">560</a>	<code>\l_block_tmpa_skip</code> <a href="#">389</a> , <a href="#">390</a> , <a href="#">391</a> , <a href="#">677</a>
<code>\__block_label_format:n</code> .....	<code>\l_block_topsepadd_skip</code> .....
..... <a href="#">38</a> , <a href="#">520</a> , <a href="#">604</a> , <a href="#">610</a>	..... <a href="#">27</a> , <a href="#">283</a> , <a href="#">332</a> , <a href="#">335</a> , <a href="#">350</a> , <a href="#">678</a>
<code>\l_block_label_given_tl</code> .....	<code>\l_block_unused_blockenv_keys_</code> -
..... <a href="#">36</a> , <a href="#">513</a> , <a href="#">535</a> , <a href="#">544</a> , <a href="#">557</a>	<code>tl</code> .. <a href="#">24</a> , <a href="#">25</a> , <a href="#">227</a> , <a href="#">241</a> , <a href="#">242</a> , <a href="#">244</a> , <a href="#">250</a>
<code>\__block_label_ref:n</code> .....	<code>block listblock-1 (instance)</code> .....
..... <a href="#">518</a>	..... <a href="#">1430</a>
<code>\l_block_label_strut_bool</code> <a href="#">521</a> , <a href="#">612</a>	<code>block listblock-2 (instance)</code> .....
<code>\g_block_labels_box</code> .....	..... <a href="#">1430</a>
.. <a href="#">36</a> , <a href="#">38</a> , <a href="#">381</a> , <a href="#">384</a> , <a href="#">584</a> , <a href="#">586</a> , <a href="#">601</a> , <a href="#">660</a>	<code>block listblock-3 (instance)</code> .....
<code>\l_block_legacy_env_params_tl</code> .	..... <a href="#">1430</a>
..... <a href="#">56</a> , <a href="#">1054</a> , <a href="#">1070</a> , <a href="#">1079</a>	<code>block listblock-4 (instance)</code> .....
<code>\l_block_legacy_support_bool</code> ..	..... <a href="#">1430</a>
..... <a href="#">445</a> , <a href="#">613</a>	<code>block listblock-5 (instance)</code> .....
	..... <a href="#">1430</a>
	<code>block listblock-6 (instance)</code> .....
	..... <a href="#">1430</a>
	<code>block quotationblock-1 (instance)</code> ..
	..... <a href="#">1415</a>
	<code>block quotationblock-2 (instance)</code> ..
	..... <a href="#">1415</a>

block quotationblock-3 (instance) ..	<a href="#">1415</a>
block quotationblock-4 (instance) ..	<a href="#">1415</a>
block quotationblock-5 (instance) ..	<a href="#">1415</a>
block quotationblock-6 (instance) ..	<a href="#">1415</a>
block quoteblock-1 (instance) .....	<a href="#">1408</a>
block quoteblock-2 (instance) .....	<a href="#">1408</a>
block quoteblock-3 (instance) .....	<a href="#">1408</a>
block quoteblock-4 (instance) .....	<a href="#">1408</a>
block quoteblock-5 (instance) .....	<a href="#">1408</a>
block quoteblock-6 (instance) .....	<a href="#">1408</a>
block theoremblock-0 (instance) ...	<a href="#">1422</a>
block verbatimblock-0 (instance) ...	<a href="#">1397</a>
block verbatimblock-1 (instance) ...	<a href="#">1397</a>
block verbatimblock-2 (instance) ...	<a href="#">1397</a>
block verbatimblock-3 (instance) ...	<a href="#">1397</a>
block verbatimblock-4 (instance) ...	<a href="#">1397</a>
block verbatimblock-5 (instance) ...	<a href="#">1397</a>
block verbatimblock-6 (instance) ...	<a href="#">1397</a>
block/endp (socket) .....	<a href="#">293</a>
block/list/label (socket) .....	<a href="#">621</a>
blockenv (templatetype) .....	<a href="#">61</a>
blockenv center (instance) .....	<a href="#">1215</a>
blockenv description (instance) ...	<a href="#">1352</a>
blockenv display (template) ....	<a href="#">66</a> , <a href="#">169</a>
blockenv displayblock (instance) ...	<a href="#">1185</a>
blockenv displayblockflattened (in- stance) .....	<a href="#">1202</a>
blockenv enumerate (instance) .....	<a href="#">1339</a>
blockenv flushleft (instance) .....	<a href="#">1229</a>
blockenv flushright (instance) ....	<a href="#">1243</a>
blockenv itemize (instance) .....	<a href="#">1326</a>
blockenv list (instance) .....	<a href="#">1364</a>
blockenv quotation (instance) .....	<a href="#">1257</a>
blockenv quote (instance) .....	<a href="#">1269</a>
blockenv theorem (instance) .....	<a href="#">1281</a>
blockenv verbatim (instance) .....	<a href="#">1292</a>
blockenv verbatim* (instance) .....	<a href="#">1309</a>
blockpar internal commands:	
<code>\__blockpar</code> .....	<a href="#">823</a>
bool commands:	
<code>\bool_gset_false:N</code> .....	<a href="#">154</a>
<code>\bool_gset_true:N</code> .....	<a href="#">149</a>
<code>\bool_if:NTF</code> .....	<a href="#">37</a> , <a href="#">42</a> , <a href="#">46</a> , <a href="#">160</a> , <a href="#">162</a> , <a href="#">195</a> , <a href="#">207</a> , <a href="#">259</a> , <a href="#">479</a> , <a href="#">547</a> , <a href="#">590</a> , <a href="#">591</a> , <a href="#">612</a> , <a href="#">613</a> , <a href="#">741</a> , <a href="#">755</a> , <a href="#">787</a> , <a href="#">789</a>
<code>\bool_if:nTF</code> .....	<a href="#">558</a>
<code>\bool_lazy_and:nnTF</code> .....	<a href="#">838</a>
<code>\bool_lazy_or:nnTF</code> .....	<a href="#">1016</a>
<code>\bool_new:N</code> .....	<a href="#">144</a> , <a href="#">603</a>
<code>\bool_set_false:N</code> .....	<a href="#">583</a>
<code>\bool_set_true:N</code> .....	<a href="#">582</a>
box commands:	
<code>\box_if_empty:N</code> .....	<a href="#">658</a>
<code>\box_new:N</code> .....	<a href="#">601</a> , <a href="#">602</a>
<code>\box_use_drop:N</code> .....	<a href="#">574</a> , <a href="#">579</a> , <a href="#">660</a>
<code>\box_wd:N</code> .....	<a href="#">563</a> , <a href="#">567</a> , <a href="#">581</a>
<code>\break</code> .....	<a href="#">591</a>
<b>C</b>	
center (env.) .....	<a href="#">971</a>
<code>\centering</code> .....	<a href="#">1523</a>
<code>\clubpenalty</code> .....	<a href="#">17</a> , <a href="#">667</a> , <a href="#">670</a>
color commands:	
<code>\color_select:n</code> .....	<a href="#">743</a>
cs commands:	
<code>\cs_generate_variant:Nn</code> .....	<a href="#">143</a>
<code>\cs_gset_protected:Npx</code> ....	<a href="#">159</a> , <a href="#">161</a>
<code>\cs_if_free:N</code> .....	<a href="#">684</a>
<code>\cs_new:Npn</code> .....	<a href="#">167</a> , <a href="#">257</a> , <a href="#">288</a> , <a href="#">290</a> , <a href="#">411</a> , <a href="#">415</a> , <a href="#">419</a> , <a href="#">426</a> , <a href="#">674</a> , <a href="#">837</a> , <a href="#">843</a> , <a href="#">846</a> , <a href="#">889</a> , <a href="#">911</a> , <a href="#">920</a> , <a href="#">932</a> , <a href="#">946</a> , <a href="#">1063</a>
<code>\cs_new_eq:NN</code> .....	<a href="#">142</a> , <a href="#">145</a> , <a href="#">146</a> , <a href="#">425</a> , <a href="#">648</a> , <a href="#">714</a> , <a href="#">715</a>
<code>\cs_new_protected:Npn</code> .....	<a href="#">139</a> , <a href="#">147</a> , <a href="#">152</a> , <a href="#">157</a> , <a href="#">164</a> , <a href="#">165</a> , <a href="#">604</a> , <a href="#">651</a> , <a href="#">704</a>
<code>\cs_set:Npe</code> .....	<a href="#">33</a> , <a href="#">429</a> , <a href="#">454</a>
<code>\cs_set:Npn</code> ....	<a href="#">716</a> , <a href="#">736</a> , <a href="#">872</a> , <a href="#">879</a> , <a href="#">880</a>
<code>\cs_set_eq:NN</code> .....	<a href="#">428</a> , <a href="#">451</a> , <a href="#">490</a> , <a href="#">598</a> , <a href="#">671</a> , <a href="#">822</a> , <a href="#">823</a> , <a href="#">824</a> , <a href="#">912</a> , <a href="#">914</a> , <a href="#">921</a> , <a href="#">923</a> , <a href="#">925</a> , <a href="#">926</a> , <a href="#">934</a> , <a href="#">936</a> , <a href="#">938</a> , <a href="#">939</a> , <a href="#">948</a> , <a href="#">950</a> , <a href="#">952</a> , <a href="#">953</a> , <a href="#">954</a> , <a href="#">955</a> , <a href="#">1086</a>
<code>\cs_set_protected:Npn</code> .....	<a href="#">799</a>
<code>\csname</code> .....	<a href="#">832</a> , <a href="#">1093</a> , <a href="#">1109</a>
<b>D</b>	
<code>\DebugBlocksOff</code> .....	<a href="#">15</a> , <a href="#">16</a> , <a href="#">164</a>
<code>\DebugBlocksOn</code> .....	<a href="#">15</a> , <a href="#">164</a>
<code>\DeclareHookRule</code> .....	<a href="#">650</a>
<code>\DeclareInstance</code> .....	<a href="#">1185</a> , <a href="#">1202</a> , <a href="#">1215</a> , <a href="#">1229</a> , <a href="#">1243</a> , <a href="#">1257</a> , <a href="#">1269</a> , <a href="#">1281</a> , <a href="#">1292</a> , <a href="#">1309</a> , <a href="#">1326</a> , <a href="#">1339</a> , <a href="#">1352</a> , <a href="#">1364</a> , <a href="#">1377</a> , <a href="#">1397</a> , <a href="#">1408</a> , <a href="#">1415</a> , <a href="#">1422</a> , <a href="#">1430</a> , <a href="#">1443</a> , <a href="#">1444</a> , <a href="#">1445</a> , <a href="#">1446</a> , <a href="#">1447</a> , <a href="#">1448</a> , <a href="#">1449</a> , <a href="#">1450</a> , <a href="#">1451</a> , <a href="#">1452</a> , <a href="#">1454</a> , <a href="#">1456</a> , <a href="#">1458</a> , <a href="#">1460</a> , <a href="#">1464</a> , <a href="#">1465</a> , <a href="#">1467</a> , <a href="#">1479</a> , <a href="#">1490</a> , <a href="#">1501</a> , <a href="#">1512</a>
<code>\DeclareInstanceCopy</code> .....	<a href="#">1391</a> , <a href="#">1392</a> , <a href="#">1393</a> , <a href="#">1394</a> , <a href="#">1395</a> , <a href="#">1396</a> , <a href="#">1402</a> , <a href="#">1403</a> , <a href="#">1404</a> , <a href="#">1405</a> , <a href="#">1406</a> , <a href="#">1407</a> , <a href="#">1410</a> , <a href="#">1411</a> , <a href="#">1412</a> , <a href="#">1413</a> , <a href="#">1414</a> , <a href="#">1417</a> , <a href="#">1418</a> , <a href="#">1419</a> , <a href="#">1420</a> , <a href="#">1421</a> , <a href="#">1428</a> , <a href="#">1429</a>
<code>\DeclareRobustCommand</code> .....	<a href="#">1523</a> , <a href="#">1524</a> , <a href="#">1525</a> , <a href="#">1526</a>
<code>\DeclareTemplateCode</code> .....	<a href="#">169</a> , <a href="#">299</a> , <a href="#">315</a> , <a href="#">432</a> , <a href="#">514</a>
<code>\DeclareTemplateInterface</code> .....	<a href="#">66</a> , <a href="#">83</a> , <a href="#">99</a> , <a href="#">111</a> , <a href="#">125</a>
<code>\def</code> .....	<a href="#">9</a> , <a href="#">10</a> , <a href="#">59</a> , <a href="#">60</a> , <a href="#">625</a> , <a href="#">779</a> , <a href="#">825</a> , <a href="#">828</a> , <a href="#">829</a> , <a href="#">871</a> , <a href="#">998</a> , <a href="#">1001</a> , <a href="#">1021</a> , <a href="#">1128</a> , <a href="#">1132</a> , <a href="#">1133</a> , <a href="#">1157</a> , <a href="#">1184</a>

default (plug) . . . . .	<a href="#">35</a> , <a href="#">625</a> , <a href="#">753</a> , <a href="#">784</a> , <a href="#">906</a>
description (env.) . . . . .	<a href="#">1023</a>
\detokenize . . . . .	<a href="#">291</a> , <a href="#">413</a> , <a href="#">417</a> , <a href="#">421</a>
dim commands:	
\dim_add:Nn . . . . .	<a href="#">374</a> , <a href="#">375</a>
\dim_compare:nNnTF . . . . .	<a href="#">277</a> , <a href="#">566</a> , <a href="#">581</a> , <a href="#">596</a>
\dim_compare_p:n . . . . .	<a href="#">562</a>
\dim_set_eq:NN . . . . .	<a href="#">373</a> , <a href="#">597</a>
\dim_zero:N . . . . .	<a href="#">505</a> , <a href="#">506</a> , <a href="#">507</a> , <a href="#">508</a> , <a href="#">509</a> , <a href="#">1064</a> , <a href="#">1065</a> , <a href="#">1066</a> , <a href="#">1084</a> , <a href="#">1085</a>
\c_zero_dim . . . . .	<a href="#">277</a> , <a href="#">808</a>
displayblock (env.) . . . . .	<a href="#">964</a>
displayblockflattened (env.) . . . . .	<a href="#">967</a>
\do . . . . .	<a href="#">1008</a>
\dospecials . . . . .	<a href="#">1008</a>
<b>E</b>	
\edef . . . . .	<a href="#">830</a>
\else . . . . .	<a href="#">1004</a>
\end . . . . .	<a href="#">269</a>
\endblockenv . . . . .	<a href="#">257</a> , <a href="#">966</a> , <a href="#">969</a> , <a href="#">973</a> , <a href="#">976</a> , <a href="#">979</a> , <a href="#">984</a> , <a href="#">987</a> , <a href="#">992</a> , <a href="#">995</a> , <a href="#">1026</a> , <a href="#">1029</a> , <a href="#">1032</a> , <a href="#">1049</a> , <a href="#">1061</a> , <a href="#">1089</a> , <a href="#">1184</a>
endblockenv . . . . .	<a href="#">15</a>
\endcsname . . . . .	<a href="#">832</a> , <a href="#">1093</a> , <a href="#">1109</a>
\endgraf . . . . .	<a href="#">824</a>
enumerate (env.) . . . . .	<a href="#">1023</a>
environments:	
center . . . . .	<a href="#">971</a>
description . . . . .	<a href="#">1023</a>
displayblock . . . . .	<a href="#">964</a>
displayblockflattened . . . . .	<a href="#">967</a>
enumerate . . . . .	<a href="#">1023</a>
flushleft . . . . .	<a href="#">971</a>
flushright . . . . .	<a href="#">971</a>
itemize . . . . .	<a href="#">1023</a>
list . . . . .	<a href="#">1051</a>
quotation . . . . .	<a href="#">981</a>
quote . . . . .	<a href="#">981</a>
trivlist . . . . .	<a href="#">1080</a>
verbatim . . . . .	<a href="#">989</a>
verbatim* . . . . .	<a href="#">989</a>
verse . . . . .	<a href="#">1034</a>
\everypar . . . . .	<a href="#">16</a> , <a href="#">20</a> , <a href="#">23</a> , <a href="#">24</a> , <a href="#">1010</a>
exp commands:	
\exp_after:wN . . . . .	<a href="#">571</a> , <a href="#">575</a>
\exp_args:Ne . . . . .	<a href="#">230</a>
\exp_args:Nee . . . . .	<a href="#">223</a> , <a href="#">234</a>
\exp_not:n . . . . .	<a href="#">33</a> , <a href="#">430</a> , <a href="#">456</a> , <a href="#">458</a>
\expandafter . . . . .	<a href="#">1010</a> , <a href="#">1093</a> , <a href="#">1109</a>
\ExplSyntaxOn . . . . .	<a href="#">8</a>
<b>F</b>	
\fi . . . . .	<a href="#">15</a> , <a href="#">28</a> , <a href="#">796</a> , <a href="#">1006</a> , <a href="#">1007</a>
fi commands:	
\fi: . . . . .	<a href="#">809</a>
\finalhyphendemerits . . . . .	<a href="#">307</a>
flushleft (env.) . . . . .	<a href="#">971</a>
flushright (env.) . . . . .	<a href="#">971</a>
\frenchspacing . . . . .	<a href="#">1011</a>
<b>G</b>	
\global . . . . .	<a href="#">59</a> , <a href="#">60</a> , <a href="#">1123</a> , <a href="#">1124</a>
group commands:	
\group_begin: . . . . .	<a href="#">1139</a> , <a href="#">1163</a>
\group_end: . . . . .	<a href="#">1149</a> , <a href="#">1176</a>
<b>H</b>	
hbox commands:	
\hbox_gset:Nn . . . . .	<a href="#">381</a> , <a href="#">584</a>
\hbox_set:Nn . . . . .	<a href="#">578</a> , <a href="#">606</a>
\hbox_set_to_wd:Nnn . . . . .	<a href="#">569</a>
\hbox_unpack_drop:N . . . . .	<a href="#">384</a> , <a href="#">573</a> , <a href="#">586</a> , <a href="#">588</a>
\hfil . . . . .	<a href="#">591</a>
hook commands:	
\hook_use:n . . . . .	<a href="#">804</a> , <a href="#">811</a>
Hooks:	
para/begin . . . . .	<a href="#">38</a> , <a href="#">40</a>
\hskip . . . . .	<a href="#">1154</a> , <a href="#">1181</a>
\hss . . . . .	<a href="#">527</a> , <a href="#">528</a> , <a href="#">529</a>
<b>I</b>	
if commands:	
\if_int_compare:w . . . . .	<a href="#">807</a>
\iffalse . . . . .	<a href="#">59</a>
\ifhmode . . . . .	<a href="#">1006</a>
\IfNoValueTF . . . . .	<a href="#">1098</a>
\iftrue . . . . .	<a href="#">60</a>
\ifvmode . . . . .	<a href="#">786</a>
\ignorespaces . . . . .	<a href="#">81</a> , <a href="#">700</a> , <a href="#">1155</a> , <a href="#">1182</a> , <a href="#">1200</a>
\indent . . . . .	<a href="#">706</a>
instances:	
block displayblock-0 . . . . .	<a href="#">1377</a>
block displayblock-1 . . . . .	<a href="#">1377</a>
block displayblock-2 . . . . .	<a href="#">1377</a>
block displayblock-3 . . . . .	<a href="#">1377</a>
block displayblock-4 . . . . .	<a href="#">1377</a>
block displayblock-5 . . . . .	<a href="#">1377</a>
block displayblock-6 . . . . .	<a href="#">1377</a>
block listblock-1 . . . . .	<a href="#">1430</a>
block listblock-2 . . . . .	<a href="#">1430</a>
block listblock-3 . . . . .	<a href="#">1430</a>
block listblock-4 . . . . .	<a href="#">1430</a>
block listblock-5 . . . . .	<a href="#">1430</a>
block listblock-6 . . . . .	<a href="#">1430</a>
block quotationblock-1 . . . . .	<a href="#">1415</a>
block quotationblock-2 . . . . .	<a href="#">1415</a>
block quotationblock-3 . . . . .	<a href="#">1415</a>

block quotationblock-4	1415	\int_incr:N	192, 197, 205, 361
block quotationblock-5	1415	\int_new:N	253
block quotationblock-6	1415	\int_set:Nn	667
block quoteblock-1	1408	\int_set_eq:NN	670
block quoteblock-2	1408	\int_to_roman:n	214
block quoteblock-3	1408	\int_use:N	224, 239, 743
block quoteblock-4	1408	\int_zero:N	356
block quoteblock-5	1408	\c_zero_int	662
block quoteblock-6	1408	\interlinepenalty	1003, 1006
block theoremblock-0	1422	item (templatetype)	61
block verbatimblock-0	1397	\item	15, 36, 541, 630, 680, 706, 1047
block verbatimblock-1	1397	item basic (instance)	1465
block verbatimblock-2	1397	item description (instance)	1465
block verbatimblock-3	1397	item std (template)	125, 512
block verbatimblock-4	1397	\itemindent	120, 442, 587, 658, 1066
block verbatimblock-5	1397	itemize (env.)	1023
block verbatimblock-6	1397	\itemsep	90, 118, 322, 439, 505, 712, 1384, 1436
blockenv center	1215	\itshape	1153, 1180
blockenv description	1352	<b>J</b>	
blockenv displayblock	1185	\justifying	1523
blockenv displayblockflattened	1202	<b>K</b>	
blockenv enumerate	1339	\kern	658
blockenv flushleft	1229	kernel internal commands:	
blockenv flushright	1243	\_kernel_displayblock_begin:	..
blockenv itemize	1326	47-49, 377, 411, 411, 916, 925, 938, 952	
blockenv list	1364	\_kernel_displayblock_beginpar_-	
blockenv quotation	1257	hmode:w	43,
blockenv quote	1269	340, 411, 415, 912, 921, 934, 948	
blockenv theorem	1281	\_kernel_displayblock_beginpar_-	
blockenv verbatim	1292	vmode	43
blockenv verbatim*	1309	\_kernel_displayblock_beginpar_-	
item basic	1465	vmode	..
item description	1465	336, 411, 419, 914, 923, 936, 950	
list description	1464	\_kernel_displayblock_end:	..
list enumerate-1	1452	47-49, 270, 290, 290, 917, 926, 939, 953	
list enumerate-2	1452	\_kernel_list_item_begin:	..
list enumerate-3	1452	48, 692, 710, 714, 714, 954	
list enumerate-4	1452	\_kernel_list_item_end:	..
list itemize-1	1448	48, 709, 714, 715, 955	
list itemize-2	1448	\_kernel_list_label_after:	..
list itemize-3	1448	661, 837, 837	
list itemize-4	1448	keys commands:	
list legacy	1460	\_keys_define:mn	35, 405, 502, 512
para center	1479	\KeyValue	88, 89,
para justify	1512	128, 1382, 1383, 1409, 1416, 1434, 1435	
para raggedleft	1501	<b>L</b>	
para raggedright	1490	\labelenumi	1453
int commands:		\labelenumii	1455
\_int_compare:nNnTF	..	\labelenumiii	1457
190, 202, 209, 362, 470, 724		\labelenumiv	1459
\_int_gdecr:N	260		
\_int_gincr:N	213		
\_int_gset:Nn	472, 481		
\_int_if_exist:NTF	251		

<code>\labelitemi</code> .....	1448	mode commands:	
<code>\labelitemii</code> .....	1449	<code>\mode_if_horizontal:TF</code> .....	
<code>\labelitemiii</code> .....	1450	.....	267, 707, 801, 806
<code>\labelitemiv</code> .....	1451	<code>\mode_if_inner:TF</code> .....	802
<code>\labelsep</code> ..	122, 444, 587, 589, 1154, 1181	<code>\mode_if_vertical:TF</code> .....	333
<code>\labelwidth</code>	121, 443, 567, 569, 581, 587, 1085	<code>\mode_leave_vertical:</code> .....	263, 331
<code>\language</code> .....	999	msg commands:	
<code>\lastbox</code> .....	23	<code>\msg_error:nmmn</code> .....	461, 540, 814
<code>\LBody</code> .....	625, 645	<code>\msg_new:nmmn</code> .....	493
<code>\leavevmode</code> .....	1003		
<code>\leftmargin</code> .....	93, 326, 374,		
	375, 383, 385, 1043, 1044, 1084, 1439	<b>N</b>	
<code>\leftskip</code> .....	303, 353	<code>\newcommand</code> .....	1015
legacy commands:		<code>\newcounter</code> .....	255
<code>\legacy_if:nTF</code> .....	39, 261,	<code>\NewDocumentEnvironment</code> .....	964, 967
	266, 273, 274, 331, 342, 348, 359,	<code>\newif</code> .....	58
	378, 387, 396, 655, 664, 675, 690,	<code>\newline</code> .....	592
	705, 718, 719, 848, 881, 890, 892, 1071	<code>\NewTaggingSocket</code> .	33, 623, 751, 782, 904
<code>\legacy_if_gset_false:n</code> .....		<code>\NewTaggingSocketPlug</code>	35, 626, 753, 784, 906
.....	264, 271, 654, 657, 666, 693, 721	<code>\NewTemplateType</code> .....	61, 62, 63, 64, 65
<code>\legacy_if_gset_true:n</code> .....		<code>\newtheorem</code> .....	11, 15, 1091
.....	27, 284, 489, 699	<code>\nobreak</code> .....	591
<code>\legacy_if_set_false:n</code> .....		<code>\nobreakspace</code> .....	1021
.....	221, 349, 380, 653, 1068	<code>\noexpand</code> .....	541, 1109
<code>\legacy_if_set_true:n</code> .....	344, 345	<code>\noindent</code> .....	1136, 1160
<code>\legacymarksetcode</code> .	11, 15, 1063, 1371	<code>\normalfont</code> .....	1469
<code>\legacymarksetup</code>	15, 997, 1305, 1322	NOT commands:	
<code>\let</code> ....	59, 60, 916, 917, 1008, 1037, 1069	<code>\NOT_IMPLEMENTED</code> .....	530
<code>\linewidth</code> .....	374, 376, 561, 563	<code>\null</code> .....	1003
<code>list (env.)</code> .....	<u>1051</u>		
<code>list (templatetype)</code> .....	<u>61</u>	<b>O</b>	
<code>\list</code> .....	1082	<code>\obeylines</code> .....	1009
<code>list description (instance)</code> .....	<u>1464</u>	<code>off (plug)</code> .....	<u>27</u> , <u>27</u> , <u>294</u>
<code>list enumerate-1 (instance)</code> .....	<u>1452</u>	<code>on (plug)</code> .....	<u>27</u> , <u>27</u> , <u>294</u>
<code>list enumerate-2 (instance)</code> .....	<u>1452</u>		
<code>list enumerate-3 (instance)</code> .....	<u>1452</u>	<b>P</b>	
<code>list enumerate-4 (instance)</code> .....	<u>1452</u>	<code>\par</code> .....	9, 16, 10,
<code>list itemize-1 (instance)</code> .....	<u>1448</u>		21, 268, 340, 706, 708, 747, 822, 1001
<code>list itemize-2 (instance)</code> .....	<u>1448</u>	par commands:	
<code>list itemize-3 (instance)</code> .....	<u>1448</u>	<code>\par_end:</code> .....	17
<code>list itemize-4 (instance)</code> .....	<u>1448</u>	par internal commands:	
<code>list legacy (instance)</code> .....	<u>1460</u>	<code>\l__par_fixed_word_spaces_bool</code> .	306
<code>list std (template)</code> .....	111, 432	<code>\l__par_start_skip</code> .....	302
<code>\listparindent</code> .....	409, 596, 597, 1064	<code>para (templatetype)</code> .....	<u>61</u>
<code>\LItag</code> .....	871, 879	<code>para center (instance)</code> .....	<u>1479</u>
<code>\tlabblockdate</code> .....	4, 1531	para commands:	
<code>\tlabblockversion</code> .....	4, 1531	<code>\para_end:</code> .....	16,
			29, 45, 364, 368, 799, 799, 822, 823, 824
		<code>\g_para_indent_box</code> .....	658
		<code>\para_omit_indent:</code> .....	659
<b>M</b>		<code>para justify (instance)</code> .....	<u>1512</u>
<code>\makelabel</code> .....	614, 1069, 1086	<code>para raggedleft (instance)</code> .....	<u>1501</u>
<code>\MakeLinkTarget</code> .	550, 552, 557, 1137, 1161	<code>para raggedright (instance)</code> .....	<u>1490</u>
math internal commands:		<code>para std (template)</code> .....	99, 299
<code>\__math_tag_dollardollar_-</code>		<code>para/begin (hook)</code> .....	38, 40
<code>display_end:</code> .....	16		

<code>para/begin</code>	16	<code>\skip_set_eq:NN</code>	350, 354, 355, 371, 372, 595
<code>\PARALABEL</code>	779, 1152, 1179	<code>\skip_vertical:n</code>	13, 279, 280, 390, 391
<code>\parfillskip</code>	305, 355	<code>\skip_zero:N</code>	353
<code>\parindent</code>	102, 301, 373, 597, 1425, 1515	<code>\l_tmpa_skip</code>	276, 277, 279, 280
<code>\parsep</code>	87, 319, 372, 440, 506, 595, 1381, 1433	socket commands:	
<code>\parskip</code>	16, 280, 352, 371, 372, 392, 397, 401, 1426	<code>\socket_assign_plug:nn</code>	298, 918, 927, 940, 956
<code>\partopsep</code>	86, 318, 335, 408, 1380, 1432	<code>\socket_if_exist:nTF</code>	31, 621, 749, 780, 902
<code>\pdfspaces</code>	1021	<code>\socket_new:nn</code>	293
<code>\penalty</code>	662, 1003, 1006	<code>\socket_new_plug:nnn</code>	294, 296
Plugs:		<code>\socket_use:n</code>	286
<code>default</code>	35, 625, 753, 784, 906	Sockets:	
<code>off</code>	27, 27, 294	<code>block/endpe</code>	293
<code>on</code>	27, 27, 294	<code>block/list/label</code>	621
prg commands:		<code>tagsupport/@doendpe</code>	31
<code>\prg_do_nothing:</code>	648, 671, 714, 715, 916, 917, 922, 924	<code>tagsupport/block/recipe</code>	902
<code>\protected</code>	825	<code>tagsupport/block/startpara/direct</code>	749
<code>\ProvidesFile</code>	1530	<code>tagsupport/kernel/endpe/vmode</code>	780
<code>\ProvidesPackage</code>	3	<code>\space</code>	4, 49, 462, 498, 1531
		str commands:	
<b>Q</b>		<code>\str_if_eq:nnTF</code>	1095
<code>quotation (env.)</code>	981	<code>\string</code>	686
<code>quote (env.)</code>	981	<code>\strut</code>	612
		sys commands:	
<b>R</b>		<code>\sys_if_engine luatex_p:</code>	1017
<code>\raggedleft</code>	1523	<code>\sys_if_output_dvi_p:</code>	1018
<code>\raggedright</code>	1523		
<code>\relax</code>	527, 529, 1047	<b>T</b>	
<code>\RemoveFromHook</code>	777	tag commands:	
<code>\renewcommand</code>	1020	<code>\tag_if_active_p:</code>	838
<code>\RenewDocumentCommand</code>	681, 1091	<code>\tag_mc_begin:n</code>	634, 742, 773, 1141, 1145, 1165, 1169, 1173
<code>\RenewDocumentEnvironment</code>	971, 974, 977, 982, 985, 990, 993, 1024, 1027, 1030, 1035, 1052, 1081	<code>\tag_mc_end:</code>	738, 744, 1143, 1147, 1167, 1171, 1175
<code>\RequirePackage</code>	6, 7, 1533	<code>\tag_socket_use:n</code>	18
<code>\rightmargin</code>	94, 325, 374, 1065, 1388, 1440	<code>\tag_socket_use:nn</code>	840, 1152, 1179
<code>\rightskip</code>	304, 313, 354	<code>\tag_socket_use:nnn</code>	608
<code>\rlap</code>	743	<code>\tag_struct_begin:n</code>	727, 758, 767, 1138, 1144, 1162, 1168
		<code>\tag_struct_end:</code>	52, 746, 792, 1148, 1150, 1172, 1177
<b>S</b>		tag internal commands:	
scan commands:		<code>\l__tag_block_flattened_level_-int</code>	22, 190, 192, 197, 251, 724
<code>\scan_stop:</code>	800	<code>__tag_check_para_begin_show:nn</code>	772
<code>\setbox</code>	23	<code>__tag_gincr_para_begin_int:</code>	765
<code>\setcounter</code>	256, 1376	<code>__tag_gincr_para_end_int:</code>	739
<code>\SetKnownTemplateKeys</code>	189, 330, 430, 453, 455, 537	<code>__tag_gincr_para_main_begin_-int:</code>	726, 757
<code>\SetTemplateKeys</code>	312	<code>__tag_gincr_para_main_end_int:</code>	51, 791, 850, 883, 894
skip commands:			
<code>\skip_add:Nn</code>	335, 352		
<code>\skip_eval:n</code>	397		
<code>\skip_horizontal:n</code>	383, 385, 587, 589		
<code>\skip_new:N</code>	677, 678, 679		
<code>\skip_set:Nn</code>	140, 313, 332		

<code>\l__tag_L_attr_class_tl</code> . . . . .	859,	<code>\doendpe</code> . . . . .	15–17, <a href="#">9</a>
	<a href="#">860</a> , <a href="#">876</a> , <a href="#">961</a> , <a href="#">962</a> , <a href="#">1072</a> , <a href="#">1074</a> , <a href="#">1075</a>	<code>\domathendpefalse</code> . . . . .	<a href="#">14</a> , <a href="#">27</a> , <a href="#">58</a>
<code>\l__tag_L_tag_tl</code> . . . . .	<a href="#">856</a> , <a href="#">857</a> , <a href="#">875</a> , <a href="#">958</a> , <a href="#">959</a>	<code>\domathendpetrue</code> . . . . .	<a href="#">58</a>
<code>\l__tag_para_attr_class_tl</code> . . . . .	<a href="#">309</a> , <a href="#">770</a>	<code>\@eha</code> . . . . .	<a href="#">828</a>
<code>\l__tag_para_bool</code> . . . . .	<a href="#">37</a> , <a href="#">787</a> , <a href="#">838</a>	<code>\@ehc</code> . . . . .	<a href="#">687</a>
<code>\g__tag_para_end_int</code> . . . . .	<a href="#">743</a>	<code>\@endparpenalty</code> . . . . .	<a href="#">6</a> , <a href="#">282</a> , <a href="#">324</a>
<code>\l__tag_para_flattened_bool</code> . . . . .	<a href="#">43</a> , <a href="#">46</a> , <a href="#">179</a> , <a href="#">195</a> , <a href="#">755</a> , <a href="#">789</a>	<code>\@endpefalse</code> . . . . .	<a href="#">45</a> , <a href="#">19</a> , <a href="#">25</a> , <a href="#">297</a> , <a href="#">794</a>
<code>\l__tag_para_main_attr_class_tl</code> . . . . .	<a href="#">730</a> , <a href="#">761</a>	<code>\@endpetrue</code> . . . . .	<a href="#">9</a> , <a href="#">295</a>
<code>\__tag_para_main_store_struct:</code> . . . . . .	<a href="#">732</a> , <a href="#">763</a>	<code>\@endtheorem</code> . . . . .	<a href="#">1124</a> , <a href="#">1184</a>
<code>\l__tag_para_main_tag_tl</code> . . . . .	<a href="#">49</a> , <a href="#">729</a> , <a href="#">760</a>	<code>\@enumdepth</code> . . . . .	<a href="#">10</a> , <a href="#">1349</a>
<code>\l__tag_para_show_bool</code> . . . . .	<a href="#">741</a>	<code>\@execute@begin@hook</code> . . . . .	<a href="#">831</a>
<code>\l__tag_para_tag_tl</code> . . . . .	<a href="#">769</a> , <a href="#">1012</a>	<code>\@flushglue</code> . . . . .	<a href="#">7</a> , <a href="#">106</a> , <a href="#">355</a> , <a href="#">1484</a> , <a href="#">1485</a> , <a href="#">1496</a> , <a href="#">1506</a> , <a href="#">1519</a>
<code>\tagmcbegin</code> . . . . .	<a href="#">638</a>	<code>\@ifdefinable</code> . . . . .	<a href="#">1093</a>
<code>\tagmcbend</code> . . . . .	<a href="#">641</a>	<code>\@ifnextchar</code> . . . . .	<a href="#">1131</a>
<code>\tagpdfparaOff</code> . . . . .	<a href="#">747</a> , <a href="#">1135</a> , <a href="#">1159</a>	<code>\@ifundefined</code> . . . . .	<a href="#">827</a> , <a href="#">1116</a>
<code>\tagpdfparaOn</code> . . . . .	<a href="#">747</a> , <a href="#">1151</a> , <a href="#">1178</a>	<code>\@ignorefalse</code> . . . . .	<a href="#">833</a>
<code>\tagpdfsetup</code> . . . . .	<a href="#">861</a> , <a href="#">1472</a>	<code>\@inmatherr</code> . . . . .	<a href="#">269</a> , <a href="#">683</a>
<code>\tagstructbegin</code> . . . . .	<a href="#">637</a> , <a href="#">645</a> , <a href="#">844</a> , <a href="#">873</a> , <a href="#">879</a>	<code>\@itemdepth</code> . . . . .	<a href="#">10</a> , <a href="#">1332</a>
<code>\tagstructend</code> . . . . .	<a href="#">644</a> , <a href="#">852</a> , <a href="#">854</a> , <a href="#">884</a> , <a href="#">887</a> , <a href="#">895</a> , <a href="#">898</a> , <a href="#">900</a>	<code>\@itemlabel</code> . . . . .	<a href="#">15</a> , <a href="#">31</a> , <a href="#">34</a> , <a href="#">219</a> , <a href="#">423</a> , <a href="#">487</a> , <a href="#">551</a> , <a href="#">632</a> , <a href="#">643</a> , <a href="#">1056</a> , <a href="#">1073</a>
<code>tagsupport/@doendpe (socket)</code> . . . . .	<a href="#">31</a>	<code>\@itempenalty</code> . . . . .	<a href="#">8</a> , <a href="#">441</a> , <a href="#">711</a>
<code>tagsupport/block/recipe (socket)</code> . . . . .	<a href="#">902</a>	<code>\@kernel@after@para@after</code> . . . . .	<a href="#">812</a>
<code>tagsupport/block/startpara/direct</code> (socket) . . . . .	<a href="#">749</a>	<code>\@kernel@after@para@end</code> . . . . .	<a href="#">805</a>
<code>tagsupport/kernel/endpe/vmode (socket)</code> . . . . .	<a href="#">780</a>	<code>\@kernel@refstepcounter</code> . . . . .	<a href="#">546</a> , <a href="#">1129</a>
template commands:		<code>\@labels</code> . . . . .	<a href="#">38</a>
<code>\template_debug_off:</code> . . . . .	<a href="#">165</a>	<code>\@latex@error</code> . . . . .	<a href="#">686</a> , <a href="#">828</a>
<code>\template_debug_on:</code> . . . . .	<a href="#">164</a>	<code>\@list...</code> . . . . .	<a href="#">6</a>
template types:		<code>\@listctr</code> . . . . .	<a href="#">31</a> , <a href="#">33</a> , <a href="#">34</a> , <a href="#">220</a> , <a href="#">423</a> , <a href="#">472</a> , <a href="#">478</a> , <a href="#">481</a> , <a href="#">546</a> , <a href="#">550</a> , <a href="#">552</a> , <a href="#">553</a> , <a href="#">1067</a>
<code>block</code> . . . . .	<a href="#">61</a>	<code>\@listdepth</code> . . . . .	<a href="#">6</a> , <a href="#">22</a> , <a href="#">167</a>
<code>blockenv</code> . . . . .	<a href="#">61</a>	<code>\@listi</code> . . . . .	<a href="#">5</a> , <a href="#">6</a>
<code>item</code> . . . . .	<a href="#">61</a>	<code>\@listii</code> . . . . .	<a href="#">5</a> , <a href="#">6</a>
<code>list</code> . . . . .	<a href="#">61</a>	<code>\@listvi</code> . . . . .	<a href="#">6</a>
<code>para</code> . . . . .	<a href="#">61</a>	<code>\@makeother</code> . . . . .	<a href="#">1008</a>
templates:		<code>\@mklab</code> . . . . .	<a href="#">1069</a>
<code>block display</code> . . . . .	<a href="#">83</a> , <a href="#">315</a>	<code>\@namedef</code> . . . . .	<a href="#">1123</a> , <a href="#">1124</a>
<code>blockenv display</code> . . . . .	<a href="#">66</a> , <a href="#">169</a>	<code>\@newctr</code> . . . . .	<a href="#">1106</a>
<code>item std</code> . . . . .	<a href="#">125</a> , <a href="#">512</a>	<code>\@nmblisttrue</code> . . . . .	<a href="#">477</a>
<code>list std</code> . . . . .	<a href="#">111</a> , <a href="#">432</a>	<code>\@nocounterr</code> . . . . .	<a href="#">1117</a>
<code>para std</code> . . . . .	<a href="#">99</a> , <a href="#">299</a>	<code>\@noitemerr</code> . . . . .	<a href="#">34</a> , <a href="#">48</a> , <a href="#">266</a> , <a href="#">348</a> , <a href="#">363</a> , <a href="#">675</a>
$\TeX$ and $\LaTeX\epsilon$ commands:		<code>\@noligs</code> . . . . .	<a href="#">1009</a>
<code>\@@par</code> . . . . .	<a href="#">1003</a> , <a href="#">1006</a>	<code>\@normalcr</code> . . . . .	<a href="#">7</a> , <a href="#">109</a> , <a href="#">1521</a>
<code>\@beginparpenalty</code> . . . . .	<a href="#">6</a> , <a href="#">323</a> , <a href="#">399</a>	<code>\@opargbegintheorem</code> . . . . .	<a href="#">58</a> , <a href="#">1133</a>
<code>\@begintheorem</code> . . . . .	<a href="#">15</a> , <a href="#">57</a> , <a href="#">1133</a>	<code>\@outerparskip</code> . . . . .	<a href="#">280</a> , <a href="#">371</a> , <a href="#">392</a> , <a href="#">397</a>
<code>\@beginthorem</code> . . . . .	<a href="#">58</a>	<code>\@restorepar</code> . . . . .	<a href="#">16</a>
<code>\@centercr</code> . . . . .	<a href="#">1037</a> , <a href="#">1488</a> , <a href="#">1499</a> , <a href="#">1510</a>	<code>\@rightskip</code> . . . . .	<a href="#">313</a> , <a href="#">354</a>
<code>\@clubpenalty</code> . . . . .	<a href="#">17</a> , <a href="#">670</a>	<code>\@setpar</code> . . . . .	<a href="#">357</a>
<code>\@currenvir</code> . . . . .	<a href="#">269</a> , <a href="#">829</a>	<code>\@setupverbinvisiblespace</code> . . . . .	<a href="#">15</a> , <a href="#">1015</a> , <a href="#">1306</a>
<code>\@currenvline</code> . . . . .	<a href="#">830</a>	<code>\@setupverbvisiblespace</code> . . . . .	<a href="#">1323</a>
<code>\@definecounter</code> . . . . .	<a href="#">1097</a>	<code>\@sxverbatim</code> . . . . .	<a href="#">63</a> , <a href="#">1324</a>
		<code>\@tempswafalse</code> . . . . .	<a href="#">1000</a>
		<code>\@tempswatrue</code> . . . . .	<a href="#">1005</a>



<code>\@thm</code> .....	15, 57, 58, 1123, 1127
<code>\@thmcounter</code> .....	1102, 1111
<code>\@thmcountersep</code> .....	1110
<code>\@toodeep</code> .....	204, 211
<code>\@topsep</code> .....	41, 70
<code>\@topsepadd</code> .....	41, 70
<code>\@totalleftmargin</code> .....	53, 375, 376
<code>\@vobeyspaces</code> .....	1306, 1323
<code>\@xobeysp</code> .....	1021
<code>\@xthm</code> .....	1131
<code>\@xverbatim</code> .....	1307
<code>\@ythm</code> .....	1131
<code>\endpefalse</code> .....	46
<code>\arabic</code> .....	8
<code>\begin</code> .....	15
<code>\bfseries</code> .....	58
<code>\bibitem</code> .....	48
<code>\g_block_nesting_depth_int</code> .....	72
<code>\c@maxblocklevels</code> .....	15, 210, 255
<code>\end</code> .....	16
<code>\endblockenv</code> .....	52
<code>\everypar</code> .....	41, 42
<code>\hyper@nopatch@thm</code> .....	1132
<code>\if@domathendpe</code> .....	12, 26, 58
<code>\if@endpe</code> .....	786
<code>\if@tempwa</code> .....	1002
<code>\ignorespaces</code> .....	5, 6, 25, 60
<code>\item</code> .....	71
<code>\item</code> .....	15, 26, 27, 29–32, 34–36, 38, 41, 48, 51, 52, 72, 73
<code>\itemindent</code> .....	54, 71
<code>\itemsep</code> .....	6
<code>\l@nohyphenation</code> .....	999
<code>\labelindent</code> .....	71
<code>\labelsep</code> .....	8, 71
<code>\labelwidth</code> .....	8, 36, 38, 71
<code>\leftmargin</code> .....	6, 71
<code>\leftskip</code> .....	53
<code>\legacylistsetupcode</code> .....	55
<code>\list</code> .....	56, 71
<code>\list(romannumeral)</code> .....	65, 67
<code>\listparindent</code> .....	28, 37, 38
<code>\makelabel</code> .....	8, 38, 56, 68, 73
<code>\newline</code> .....	36
<code>\newtheorem</code> .....	57
<code>\noitemerr</code> .....	26
<code>\on@line</code> .....	45, 50, 258, 652, 718, 740, 766, 830, 847, 851, 885, 896
<code>\par</code> .....	14, 16, 17, 27, 29, 34, 42, 43, 45, 49–51, 71
<code>\par@deathcycles</code> .....	356, 361, 362
<code>\parindent</code> .....	7, 38, 67
<code>\parsep</code> .....	6
<code>\parskip</code> .....	30, 67
<code>\partopsep</code> .....	6, 70
<code>\pdfspaces</code> .....	54, 63
<code>\ref</code> .....	71
<code>\refstepcounter</code> .....	57
<code>\reserved@a</code> .....	828, 829, 836
<code>\rightmargin</code> .....	6
<code>\SetKnownTemplateKeys</code> .....	31
<code>\spacefactor</code> .....	34
<code>\strut</code> .....	8, 38
<code>\textbf</code> .....	58
<code>\topsep</code> .....	6, 70
<code>\trivlist</code> .....	71
<code>\UseInstance</code> .....	41
<code>\verbatim@font</code> .....	1009
<code>\z@</code> .....	23
<code>\z@skip</code> .....	1486, 1495, 1497, 1507, 1508, 1517, 1518
tex commands:	
<code>\tex_hskip:D</code> .....	808
<code>\tex_lastnodetype:D</code> .....	807
<code>\tex_lastskip:D</code> .....	140
<code>\tex_par:D</code> .....	810, 819
<code>\tex_parshape:D</code> .....	376
<code>\tex_parskip:D</code> .....	13
<code>\tex_unskip:D</code> .....	142, 803
<code>\the</code> .....	1010
<code>\tiny</code> .....	743
tl commands:	
<code>\c_novaluel_tl</code> .....	36, 535
<code>\tl_clear:N</code> .....	219, 220
<code>\tl_gset:Nn</code> .....	1100, 1107, 1119
<code>\tl_if_blank:nTF</code> .....	546
<code>\tl_if_empty:NTF</code> .....	200, 228, 232, 237, 244, 459, 468, 485, 538, 928, 941, 957, 960, 1073
<code>\tl_if_empty:nTF</code> ..	427, 450, 632, 643
<code>\tl_if_eq:NnTF</code> .....	289
<code>\tl_if_novalue:nTF</code> ..	143, 544, 696
<code>\tl_new:N</code> .....	250, 423, 424, 600, 856, 859, 945, 1079, 1127
<code>\tl_set:Nn</code> .....	527, 528, 529, 857, 860, 929, 942, 958, 961, 1012, 1054, 1056, 1067, 1072, 1074, 1075, 1130
<code>\tl_set_eq:NN</code> .....	227, 242, 478, 487, 535, 930, 943, 959, 962
<code>\topsep</code> .....	85, 317, 332, 407, 507, 510, 1379, 1431
<code>trivlist (env.)</code> .....	1080
<code>\typeout</code> .....	162
<b>U</b>	
<code>\unpenalty</code> .....	1010
<code>\UnusedTemplateKeys</code> .....	22, 24, 226, 227, 242, 456, 459, 463, 538, 542

use commands:	968, 972, 975, 978, 983, 986, 991,
\use:N .....	214, 908
\use:n .....	615, 1086
\use_i:nn .....	571
\use_ii:nn .....	575
\use_none:n .....	145, 146
\use_none:nn .....	425, 428, 451
\usecounter .....	33
\UseHook .....	826
\UseInstance .....	223, 230, 235, 965,
	994, 1025, 1028, 1031, 1038, 1059,
	1134, 1158, 1523, 1524, 1525, 1526
\UseName .....	91,
	92, 119, 1120, 1385, 1386, 1437, 1438
\UseTaggingSocket .....	218, 818, 835
	<b>V</b>
verbatim (env.) .....	<u>989</u>
verbatim* (env.) .....	<u>989</u>
verse (env.) .....	<u>1034</u>