

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-10-10<sup>\*</sup>

©2024 by Pablo González<sup>†</sup>

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

## Abstract

This package provides enumerated list environments compatible with L<sup>A</sup>T<sub>E</sub>X tagging PDF for creating “simple exercise sheets” along with “multiple choice questions”, storing the “answers” to these in memory using `multicol` and `scontents` packages and the `l3seq` and `l3prop` modules.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>	<b>6</b>	<b>The storage system</b>	<b>11</b>
1.1	Description and usage	3	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	12
1.3	User interface	4	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	4	6.1.3	Keys for debug and checking	13
1.3.2	Public dimension	4	6.2	The command <code>\anskey</code>	13
1.3.3	Support for multicol	4	6.2.1	Keys for <code>\anskey</code>	13
1.3.4	Support for minipage	4	6.3	The environment <code>anskey*</code>	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	15
<b>2</b>	<b>The environments provided</b>	<b>5</b>	6.4.1	The <code>\item*</code> in <code>keyans</code>	15
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	16
2.3	The command <code>\item*</code>	5	6.5.2	The command <code>\anspic</code>	16
2.3.1	Keys for <code>\item*</code>	6	6.6	Printing stored content	17
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	17
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>6</b>	6.6.2	The command <code>\foreachkeyans</code>	17
<b>4</b>	<b>The command <code>\setenumextmeta</code></b>	<b>6</b>	6.6.3	The command <code>\printkeyans</code>	18
<b>5</b>	<b>The keyval system</b>	<b>7</b>	<b>7</b>	<b>Full examples</b>	<b>19</b>
5.1	Keys for label and ref	7	<b>8</b>	<b>Tagged PDF examples</b>	<b>21</b>
5.2	Keys for spaces	8	<b>9</b>	<b>The way of non-enumerated lists</b>	<b>22</b>
5.2.1	Vertical spaces	8	<b>10</b>	<b>References</b>	<b>24</b>
5.2.2	Horizontal spaces	9	<b>11</b>	<b>Change history</b>	<b>24</b>
5.3	Keys for add code	9	<b>12</b>	<b>Index of Documentation</b>	<b>25</b>
5.4	Keys for start, series and resume	10	<b>13</b>	<b>Implementation</b>	<b>27</b>
5.5	Keys for multicol	10	<b>14</b>	<b>Index of Implementation</b>	<b>142</b>
5.6	Keys for minipage	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

## Motivation and acknowledgments

Usually it is enough to use the classic `enumerate` environment to generate “simple exercise sheets” or “multiple choice questions”, the basic idea behind `enumext` is to cover three points:

1. To have a simple interface to be able to write “lists of exercises” with “answers”.
2. To have a simple interface for writing “multiple choice questions”.
3. To have a simple interface for placing “columns” and “drawings” or “tables”.

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all L<sup>A</sup>T<sub>E</sub>X team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

1. Answer given by Alan Munn in `\topsep`, `\itemsep`, `\partopsep`, `\parsep` - what do they each mean (and what about the bottom)?
2. Answer given by Enrico Gregorio in Understanding minipages - aligning at top
3. Answer given by Ulrich Diez in Different mechanics of hyperlink vs. hyperref
4. Answer given by Enrico Gregorio in Minipage and multicol, vertical alignment

<sup>\*</sup>This file describes a documentation for v1.0, last revised 2024-10-10.

<sup>†</sup>E-mail: [pablgonz@educarchile.cl](mailto:pablgonz@educarchile.cl).

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp), version 1.3 or later (<https://www.latex-project.org/lppl.txt>). The software has the status “maintained”.  
The enumext package loads and requires multicol[3] and scontents[4] packages, need to have a modern T<sub>E</sub>X distribution such as T<sub>E</sub>X Live or MiK<sub>T</sub>E<sub>X</sub>. It has been tested with the standard classes provided by L<sup>A</sup>T<sub>E</sub>X: book, report, article and letter on 10pt, 11pt and 12pt.

1 Introduction

In the L<sup>A</sup>T<sub>E</sub>X world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like exam[1] and packages like xsim[2] do the job perfectly, but they don’t always fit the basic day to day needs.

In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

1. Factor  $x^2 - 2x + 1$

2. Factor  $3x + 3y + 3z$

3. True False

(a)  $\alpha > \delta$

(b) L<sup>A</sup>T<sub>E</sub>X is cool?

4. Related to Linux
- (a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

Sometimes we are also interested in showing the “answers” along with the questions:

1. Factor  $x^2 - 2x + 1$

\* 

$(x - 1)^2$

2. Factor  $3x + 3y + 3z$

\* 

$3(x + y + z)$

3. True False

(a)  $\alpha > \delta$

\* 

False

(b) L<sup>A</sup>T<sub>E</sub>X is cool?

\* 

Very True!

4. Related to Linux
- (a) You use linux?

\* 

Yes

(b) Usually uses the package manager?

\* 

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

\* 

doesn't exist for now :(

ii. xsim

\* 

very good

iii. exsheets

\* 

obsolete

Or we are interested in referring to a specific question and its “answer”, for example:

The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.

Or we are interested in printing all the “answers”:

1.  $(x - 1)^2$

2.  $3(x + y + z)$

3. (a) False

(b) Very True!

4. (a) Yes
- ⌘ (b) Yes, dnf

⌘ (c) i. doesn't exist for now :(

⌘ ii. very good

⌘ iii. obsolete

⌘

Another very common thing to use in my work is “multiple choice questions”, for example:

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

I.  $2\alpha + 2\delta = 90^\circ$

II.  $\alpha = \delta$

III.  $\angle EDF = 45^\circ$

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

\* 3. Third type of questions

(1)  $2\alpha + 2\delta = 90^\circ$

(2)  $\angle EDF = 45^\circ$

A) value

B) value

C) value

D) value

E) value
4. Question with image and label below:

A

A)

B


B)

A

C)

A

D)



E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B
- Where what we are interested in the *label* and a “short note” that we leave as an explanation, and then print them:
- ©2024 by Pablo González L
- 2 / 157

1. B)  $x = 5$

2. D)

3. C) some note
- ※ 4. E) A duck

※ 5. D) “other note”

※

These “*simple worksheets*” or “*multiple choice questions*” appear to be easy to obtain using a combination of the `enumerate`, `minipage` and `multicols` environments, but like many things, what “*looks simple*” is not so simple.

The `enumext` package was created and designed to meet these small requirements in the creation of “*simple worksheets*” and “*multiple choice questions*”.

1.1 Description and usage

The `enumext` package defines enumerated environments using the `list` environment provided by  $\text{\LaTeX}$ , but “*does not redefine*” any internal commands associated with it such as `\list`, `\endlist` or `\item` outside of the “*scope*” in which they are defined.

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in  $\text{\TeX}$  Live and  $\text{\MiKTeX}$ , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

```
enumext.sty  >> TDS:tex/latex/enumext/
enumext.pdf  >> TDS:doc/latex/enumext/
README.md   >> TDS:doc/latex/enumext/
enumext.dtx  >> TDS:source/latex/enumext/
```

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an “*extra space*” that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment.

Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.



Figure 1: Representation of horizontal lengths in `enumitem`.

The `enumext` package does NOT provide a user interface to set the values for `\leftmargin` and `\itemindent`, instead it provides the keys `list-offset` and `list-indent` which internally set the values for `\leftmargin` and `\itemindent`. The concepts of `\leftmargin` and `\itemindent` are different in `enumext`. The figure 2 shows the visual representation of idea.

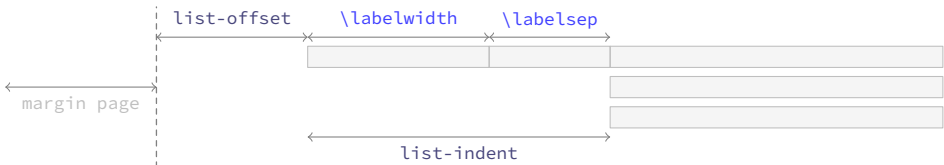


Figure 2: Representation of horizontal lengths concept in `enumext`.

In this way we reduce a *little* the amount of parameters we have to pass. With the default values of keys `list-offset`, `list-indent`, `labelwidth` and `labelsep` the lists will have the (usually) expected output for “*simple worksheets*”. The figure 3 shows the visual representation.

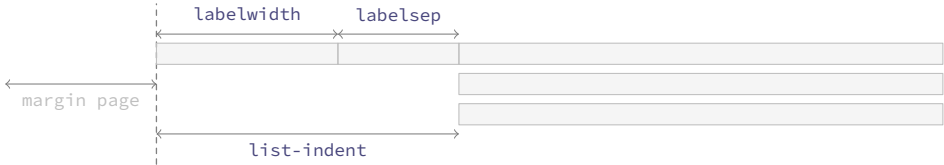


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

### 1.3 User interface

The user interface consists of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to “store content” and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print individual *stored content*, `\printkeyans` to print all *stored content*, `\miniright` for `minipage` and `\setenumext` to config all `[key = val]` options.

#### 1.3.1 Internal counters

The package `enumext` uses internally the `enumXi`, `enumXii`, `enumXiii`, `enumXiv` counters for the four nesting levels of the `enumext` environment, the `enumXv` counter for the `keyans` environment, the `enumXvi` counter for the `keyanspic` environment, the counter `enumXvii` for `enumext*` environment and the counter `enumXviii` for `keyans*` environment.

- If any package defines these counters or they are user-defined in the document, the package will return a fatal error and abort the load.

#### 1.3.2 Public dimension

The package `enumext` only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. The dimension `\itemwidth` is *rigid length* and contains the “width of the content” of each `\item` regardless of `labelwidth` and `labelsep`.

- If any package defines `\itemwidth` or they are user-defined `\itemwidth` in the document, the package will overwrite it without warning.

#### 1.3.3 Support for multicol

The package provides direct support for using the `multicol[3]` package. This allows to obtain directly a two-column output as shown in the figure 4.

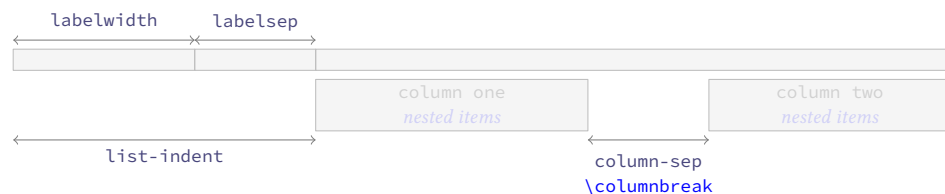


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §5.5).

- The `\columnseprule` command is not available as a key and is set to “zero” for the inner levels and the `keyans` environment. If the value of this is set inside the document, it will affect “all environments” that use the `columns` key.

#### 1.3.4 Support for minipage

The package provides direct support for `minipage` environment, this allows you to obtain an output like the one shown in figure 5.



Figure 5: Representation of the `mini-env` output for a nested level `enumext` environment.

The `minipage` environments on “left side” and “right side” is always used with “aligned on top” `[t]`. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the “left” side `minipage` environment to the “right” side one must use the command `\miniright` (see §5.6).

#### 1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem[6]` package to customize the references which is activated by the `ref` key (§5.1), the standard  $\TeX$  `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§6.1.1) when the key `save-ans` (§6.1) is active.

#### 1.3.6 Support for \footnote

This package provides an internal implementation for the `\footnote` command which is compatible with the `hyperref` package for the `enumext*` and `keyans*` environments, but will not produce the expected links, and if the `mini-env` key is used in `enumext` or `keyans` environments the output will look like the classic way they are displayed in the environment `minipage`.

The best way to solve this is to use Jean-François Burnol `footnotehyper`[9] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

```
\usepackage{footnotehyper}
\makesavenoteenv{enumext}
\makesavenoteenv{enumext*}
```

At the moment the `footnotehyper` package is not compatible with *tagged* PDF.

## 2 The environments provided

The package `enumext` provides two main list environments, the *vertical* environment `enumext` and the *horizontal* environment `enumext*`.

<code>enumext</code>	<code>\begin{enumext}[\langle keyval list \rangle]</code>	<code>\begin{enumext*}[\langle keyval list \rangle]</code>
<code>enumext*</code>	<code>\item \langle item content \rangle</code>	<code>\item \langle item content \rangle</code>
	<code>\item [\langle custom \rangle] \langle item content \rangle</code>	<code>\item [\langle custom \rangle] \langle item content \rangle</code>
	<code>\item* [\langle symbol \rangle] [\langle offset \rangle] \langle item content \rangle</code>	<code>\item* [\langle symbol \rangle] [\langle offset \rangle] \langle item content \rangle</code>
	<code>\end{enumext}</code>	<code>\end{enumext*}</code>

### 2.1 The environment `enumext`

The `enumext` is an environment that works in the same way as the standard `enumerate` environment provided by L<sup>A</sup>T<sub>E</sub>X, `\item` and `\item[\langle custom \rangle]` commands work in the usual way. The environment can be nested with at most “four levels” and the options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Example with `columns=2`

1. This text is in the first level.
- A. This text is in the fourth level.
- (a) This text is in the second level.
- X This text is in the first level.
- i. This text is in the third level.
- \* 2. This text is in the first level.

### 2.2 The environment `enumext*`

The `enumext*` is a *horizontal list environment* similar to the `enumerate*` environment provided by the `enumitem` package or `task` environment provided by the `task` package, `\item` and `\item[\langle custom \rangle]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each “*item content*” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that NOT includes `labelwidth`, `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.
- You cannot have any standard list environments like `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

Example with `columns=2`

1. This text is in the first level.
2. This text is in the first level.
- X This text is in the first level.
- \* 4. This text is in the first level.

### 2.3 The command `\item*`

<code>\item*</code>	<code>\item*</code>
	<code>\item* [\langle symbol \rangle]</code>
	<code>\item* [\langle symbol \rangle] [\langle offset \rangle]</code>

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` works like the numbered `\item`, but placing a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the `\langle offset \rangle` set by the *second optional argument*. The default values for `\langle symbol \rangle` and `\langle offset \rangle` are `\textasteriskcentered *` and the value set by `labelsep` key.

The *starred argument* “`*`” cannot be separated by spaces “`␣`” from the command, i.e. `\item*` and the *first optional argument* does “NOT” support *verbatim content*. Can be configure with the keys `item-sym*` and `item-pos*` locally in the environment or globally using `\setenumext` command (§3).

The behavior of `\item*` in the `enumext` and `enumext*` environments is NOT the same as in the `keyans` and `keyans*` environments.

2.3.1 Keys for \item\*

`item-sym*` = {<symbol>} default: \textasteriskcentered  
Sets the *symbol* to be displayed in the “left” of the box containing the current <label> set by `labelwidth` key for `\item*` in `enumext` and `enumext*`. The *symbol* can be in *text* or *math* mode, for example `item-sym*={\star}`.  
`item-pos*` = {<rigid length>} default: by levels  
Sets the *offset* between the box containing the current <label> defined by `labelwidth` key and the <symbol> set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

2.4 The command \item in enumext\*

The `\item` command for the `enumext*` environment provides an “first optional argument” `\item(<columns>)` which “joins items” between columns. Let’s consider the following examples adapted directly from the `task` package:

```
\begin{enumext*}[widest=10,columns=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item The seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
    (\the\itemwidth)
  \item The ninth
  \item[Z] The tenth (\the\itemwidth)
\end{enumext*}
```

1. The first
- \* 2. The second
3. The third
4. The fourth
- \* 5. The fifth item is way too long for this and needs three columns
6. The sixth
7. The seventh
- X 8. The eighth item is way too long for this and needs two columns (196.17749pt)
9. The ninth
- Z 10. The tenth (89.28171pt)

3 The command \setenumext

<code>\setenumext</code>	<code>\setenumext{&lt;key = val&gt;}</code>	<code>\setenumext[&lt;keyans*&gt;]{&lt;key = val&gt;}</code>
	<code>\setenumext[&lt;enumext, level&gt;]{&lt;key = val&gt;}</code>	<code>\setenumext[&lt;print, level&gt;]{&lt;key = val&gt;}</code>
	<code>\setenumext[&lt;enumext*&gt;]{&lt;key = val&gt;}</code>	<code>\setenumext[&lt;print, *&gt;]{&lt;key = val&gt;}</code>
	<code>\setenumext[&lt;keyans&gt;]{&lt;key = val&gt;}</code>	<code>\setenumext[&lt;print*&gt;]{&lt;key = val&gt;}</code>

The command `\setenumext` sets the <keys> on a global basis for environments `enumext`, `enumext*`, `keyans`, `keyans*` and the `\printkeyans` command. It can be used both in the preamble and in the body of the document as many times as desired.

The <keys> set in the *optional argument* of environments and commands have the *highest precedence*, overriding both options passed by `\setenumext`. If the *optional argument* is not passed, the first level of the environment `enumext` will be taken by default.

- The key `save-ans` that activate the “storage system” must NOT be passed through this command and must be passed directly in the *optional argument* of the “first level” of the environment in which they are executed.

4 The command \setenumextmeta

<code>\setenumextmeta</code>	<code>\setenumextmeta {&lt;key name&gt;}{&lt;key-one = val, key-two = val, ...&gt;}</code>
	<code>\setenumextmeta*{&lt;key name&gt;}{&lt;key-one = val, key-two = val, ...&gt;}</code>
	<code>\setenumextmeta [&lt;enumext*&gt;]{&lt;key name&gt;}{&lt;key-one = val, key-two = val, ...&gt;}</code>
	<code>\setenumextmeta [&lt;enumext, level&gt;]{&lt;key name&gt;}{&lt;key-one = val, key-two = val, ...&gt;}</code>

The command `\setenumextmeta` adds a new “meta-key” for the environments `enumext` and `enumext*`, the {<key name>} must be different from those defined by the package. If the *optional argument* is not passed, the new “meta-key” will be created for the “first level” of the environment `enumext`.

The *starred argument* ‘\*’ will create the new “meta-key” for the environment `enumext*` and for all levels of the environment `enumext`. For example: `\setenumextmeta*{midsep}{topsep=3pt, partopsep=0pt}` will create a new key `midsep` available for all levels of the `enumext` environment and the `enumext*` environment and we can use it like any other key so `\begin{enumext}[midsep]` and `\begin{enumext*}[midsep]` will be valid.



## 5 The keyval system

The  $\langle key = val \rangle$  system used by the `enumext` package is implemented using `l3keys` so it must be taken into consideration that those keys marked as “*value forbidden*”, that is  $\langle key \rangle$  is different from  $\langle key = \rangle$ .

All  $\langle keys \rangle$  described in this section are available for the `enumext`, `enumext*`, `keyans` and `keyans*` environments with the exception of the keys `series`, `resume`, `resume*` which are only available for the “*first level*” of the environments `enumext` and `enumext*`; and the keys `mini-right`, `mini-right*` which are only available for the `enumext*` and `keyans*` environments.

All  $\langle keys \rangle$  related to vertical or horizontal spacing accept a “*skip*” or “*dim*” expression if passed between braces, i.e. you do not need to use `\dimeval` or `\dimexpr` to perform calculations.

- It should be kept in mind that using any  $\langle key \rangle$  that sets a *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and `keyans`, `keyans*` and `keyanspic` environments.

### 5.1 Keys for label and ref

`mode-box`  $\langle value forbidden \rangle$  default: *not used*

This is a “*switch-key*” that does not receive an argument and is “*only*” available for the “*first level*” of the `enumext` environment and the `enumext*` environment. When this is set the `label`, `font`, `wrap-label` and `wrap-label*` keys are executed within `\makebox` for the `enumext` and `keyans` environments.

- This key is intended for compatibility with *tagged* PDF and is forcibly “*enabled*” when `\DocumentMetadata` is present. If you want to get the same document output whether `\DocumentMetadata` is active or not, you must enable this key.
- In the `enumext*` and `keyans*` environments `\makeLabel` are redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

`label` = {  $\langle \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \rangle$  } default: *by levels*

Sets the  $\langle label \rangle$  that will be printed at the *current level*. The default value for the first level of the environments `enumext` and `enumext*` are `\arabic*`, for second level are  $\langle \backslash alph* \rangle$ , for third level are `\roman*`, and for fourth level are `\Alph*`. For `keyans` and `keyans*` environments the default value is `\Alph*`.

- This key is intended to give the basic structure with which the  $\langle label \rangle$  will be displayed, and the form in which it is used by standard “*label and ref*” and the “*internal label and ref*” system with the `save-ref` key. You cannot use commands with  $\langle label \rangle$  as an argument, for example `\emph{\backslash alph*}` will return an error. For full customization of how  $\langle label \rangle$  is displayed use the `font`, `wrap-label` and/or `wrap-label*` keys.

`labelsep` = {  $\langle rigid length \rangle$  } default: `0.3333em`

Sets the *horizontal space* between the box containing the current  $\langle label \rangle$  defined by `label` key and the text of an item on the first line. Internally sets the value of `\labelsep` for the current level.

`labelwidth` = {  $\langle rigid length \rangle$  } default: *by label*

Sets the *width* of the box containing the current  $\langle label \rangle$  set by `label` key. Internally sets the value of `\labelwidth` for the current level. The default values are calculated by means of the *width* of a box by setting a *value* to the current counter using ‘0’ for `\arabic*`, ‘M’ for `\Alph*`, ‘m’ for `\alph*`, ‘VIII’ for `\Roman*` and ‘viii’ for `\roman*`.

`widest` = {  $\langle integer | string \rangle$  } default: *empty*

Sets the `labelwidth` key pass the  $\langle integer \rangle$  or converting the  $\langle string \rangle$  of the form `\Alph`, `\alph`, `\Roman` or `\roman` to a *value* for the current counter defined by `label` key, then calculating the *width* by means of a box. For example `widest={XXIII}` or `widest={23}` are equivalent. This key is useful when the default values of the `labelwidth` key are smaller than those actually used.

`font` = {  $\langle font commands \rangle$  } default: *empty*

Sets the *font style* for the current  $\langle label \rangle$  defined by `label` key. For example `font={\bfseries\small}`.

`align` = {  $\langle left | right | center \rangle$  } default: *left*

Sets the *aligned* of  $\langle label \rangle$  defined by `label` key on the current level in the label box.

`wrap-label` = {  $\langle code \{ \#1 \} \text{ more code} \rangle$  } default: *empty*

Wraps the *current*  $\langle label \rangle$  defined by `label` key referenced by `\{ \#1 \}`. The  $\langle code \rangle$  must be passed between braces. This key does not modify the value set by the `labelwidth` key and is applied only on `\item` and `\item*`. When using it in the `\setenumext` command it is necessary to use the *double hash* ‘`\{ \# \#1 \}`’. For example `wrap-label={\fbox{\#1}}` or you can create a command:

```
\NewDocumentCommand \labelbx { s +m }
{%
  \IfBooleanTF{\#1}
  {\strut\smash{\parbox[t]{\labelwidth}{\raggedright{\#2}}}}%
  {\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{\#2}}}}%
}
```

and then pass it through the key `wrap-label={\labelbx{\#1}}` or `wrap-label={\labelbx*{\#1}}`.

`wrap-label*` = {  $\langle code \{ \#1 \} \text{ more code} \rangle$  } default: *empty*

The same as the `wrap-label` key but also applies on `\item[custom]`.

`ref = {⟨code⟩ {⟨\alph*⟩⟨\Alph*⟩⟨\arabic*⟩⟨\roman*⟩⟨\Roman*⟩} more code}⟩` default: *empty*

Modifies the way *cross references* are displayed. The `label` key sets the default form of the *cross references*, by using this key you can define a different format, for example: `ref=\emph{⟨\alph*⟩}` is valid.

Internally it renews the command associated with each counter when it is executed, i.e., in the environment `enumext` the command `\theenumxi` is modified when the key is executed at the first level, `\theenumxii` when it is executed at the second level and `\theenumxiii` together with `\theenumxiv` when it is executed at the third and fourth levels.

- This must be kept in mind, since the values set by the `label` and `ref` keys are not cumulative by levels, so if you have used the `ref` key in the first level and then want to associate the counter with `label` or `ref` in the second level you must use the direct commands, i.e. `\arabic{enumxi}` to indicate the count of the first level instead of using `\theenumxi`.

## 5.2 Keys for spaces

`show-length = {⟨true⟩|⟨false⟩}` default: *false*

Displays on the terminal the values for *all list parameters* at the current level. For *vertical spaces* show the values of `\topsep`, `\itemsep`, `\parsep` and `\partopsep`. For *horizontal spaces* show the values of `\labelwidth`, `\labelsep`, `\itemindent`, `\listparindent` and `\leftmargin`.

### 5.2.1 Vertical spaces

`topsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

Set the *vertical space* added to both the top and bottom of the list. Internally sets the value of `\topsep` for the current level. The default value for the first level of the environments `enumext` and `enumext*` are `8.0pt` plus `2.0pt` minus `4.0pt`, for second level are `4.0pt` plus `2.0pt` minus `1.0pt`, for third and fourth level are `2.0pt` plus `1.0pt` minus `1.0pt`. For `keyans` and `keyans*` environments the default value is `4.0pt` plus `2.0pt` minus `1.0pt`.

`parsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

Set the *vertical space* between paragraphs within an item. Internally sets the value of `\parsep` for the current level. The default value for the first level of the environments `enumext` and `enumext*` are `4.0pt` plus `2.0pt` minus `1.0pt`, for second level are `2.0pt` plus `1.0pt` minus `1.0pt`, for third and fourth level are `0pt`. For `keyans` and `keyans*` environments the default value is `2.0pt` plus `1.0pt` minus `1.0pt`.

- In the `enumext*` and `keyans*` environments this value is passed to `\parskip` within the `minipage` environment where “item content” is placed.

`partopsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

Set the *vertical space* added, beyond `topsep`, to the “top” and “bottom” of the entire environment if the environment instance is preceded by a “blank line” or `\par` command. Internally sets the value of `\partopsep` for the current level. The default values for first and second level in environment `enumext` are `2.0pt` plus `1.0pt` minus `1.0pt`, for third and fourth level are `1.0pt` minus `1.0pt`. For the `keyans` environment the default value is `2.0pt` plus `1.0pt` minus `1.0pt`, and for the `keyans*` and `enumext*` environments it is available but *without* effect.

- The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T<sub>E</sub>X will enter *⟨vertical mode⟩* and apply this value to the “top” and “bottom” the environment or nested level.

`itemsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

Set the *vertical space* between items, beyond the `parsep`. Internally sets the value of `\itemsep` for the current level. The default value for the first level of the environments `enumext` and `enumext*` are `4.0pt` plus `2.0pt` minus `1.0pt`, for the rest of the levels are `2.0pt` plus `1.0pt` minus `1.0pt`. For `keyans` and `keyans*` environments the default value is `4.0pt` plus `2.0pt` minus `1.0pt`.

- In the `enumext*` and `keyans*` environments this value corresponds to the separation between rows.

`noitemsep` *⟨value forbidden⟩* default: *not used*

This is a “meta-key” that does not receive an argument. Set `itemsep` and `parsep` equal to `0pt` the entire level of environment.

`nosep` *⟨value forbidden⟩* default: *not used*

This is a “meta-key” that does not receive an argument. Sets all keys for vertical spacing equal to `0pt` the entire level of environment.

`base-fix` *⟨value forbidden⟩* default: *not used*

This is a “switch-key” that does not receive an argument available *only* for the “first level” of environment `enumext`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` and there is no material between the `\item` and the start of the environment for example `\item \begin{enumext}` within the environment `enumext*`. Internally sets the keys `topsep`, `above` and `above*` at `0pt`.



- The following *<keys>* should be used with “caution”, they are intended to be used at the “top” and “bottom” of the environment when the `columns` or `mini-env` keys do not provide adequate *vertical spaces*. The values passed can be *rubber* or *rigid* lengths, the way they are applied is the way you differ, using the star ‘\*’ *<keys>* applies `\vspace*` so that  $\text{\LaTeX}$  does *not discard* this space at page break.

`above = {<rubber length | rigid length>}` default: *not used*

Set the *extra vertical space* added, beyond `topsep`, to the top of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace` so is “discardable”.

`above* = {<rubber length | rigid length>}` default: *not used*

Set the *extra vertical space* added, beyond `topsep`, to the top of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace*` so is “not discardable”.

`below = {<rubber length | rigid length>}` default: *not used*

Set the *extra vertical space* space added, beyond `topsep`, to the bottom of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space on the “below” the environment without hindering the value of the `topsep` key. The space is added with `\vspace` so is “discardable”.

`below* = {<rubber length | rigid length>}` default: *not used*

Set the *extra vertical space* space added, beyond `topsep`, to the bottom of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space on the “below” the environment without hindering the value of the `topsep` key. The space is added with `\vspace*` so is “not discardable”.

### 5.2.2 Horizontal spaces

`list-offset = {<rigid length>}` default: `0pt`

Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.

`list-indent = {<rigid length>}` default: `labelwidth + labelsep`

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level. If `list-indent=0pt` is set in the environments `enumext` and `keyans` the *<label>* will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “common paragraph”.

- The `enumext*` and `keyans*` environments are implemented using `\makebox` and `minipage` which causes “list indent” to always be equal to the value passed to `labewdith` plus `labelsep`. Passing a value to this key is equivalent to setting the value for the `list-offset` key.

`itemindent = {<rigid length>}` default: `0pt`

Sets the extra *horizontal indentation*, beyond `labelsep`, of the “first line” off each `\item` that is not followed by a “blank line” or the `\par` command. This value must be greater than or equal to `0pt` and is applied internally using `\hspace` without modifying the value of `\itemindent`.

- This key is intended for the `enumext*` and `keyans*` environments where, by their implementation, it is not possible to adjust `labelwidth` and `list-indent` without modifying the output. If you use `enumext` or `keyans` and want to get around the *blank line* limitation or the `\par` command followed by `\item` you can modify `labelwidth` and `label-indent` and get the same effect.

`rightmargin = {<rigid length>}` default: `0pt`

Set the *horizontal space* between the right margin of the environment and the right margin of the enclosing environment, the value it takes must be greater than or equal to `0pt`. Internally sets the value of `\rightmargin` for the current level.

`listparindent = {<rigid length>}` default: `0pt`

Sets the *horizontal space* indentation, beyond `list-indent`, for second and subsequent paragraphs within a list item. Internally sets the value of `\listparindent` for the current level.

- In the `enumext*` and `keyans*` environments this value is passed to `\parindent` within the `minipage` environment where “item content” is placed.

## 5.3 Keys for add code

The following *<keys>* should be used with “caution”, they are intended to inject `{<code>}` into different parts of the defined environments. We must keep in mind that the defined environments are based on the `list` base environment provided by  $\text{\LaTeX}$  which is defined (simplified) as plain form `\list{<arg one>}{<arg two>}`. Using the `before*` key does not allow access to the `list` parameters defined by `[<key = val>]`.

`before = {<code>}` default: *not used*

Execute `{<code>}` “before” the environment starts. The `{<code>}` must be passed between braces, is executed “after” performing all calculations related to the *list parameters* in the environment and the parameters sets by `[<key = val>]` that is, in the second argument of the list after setting all the parameters `\begin{list}{<arg one>}{<arg two>}{<code>}`.

`before* = {⟨code⟩}` default: *not used*  
 Execute {⟨code⟩} “before” the environment starts. The {⟨code⟩} must be passed between braces, is executed “before” performing all calculations related to the *list parameters* and [`key = val`] sets in the environment that is, before the arguments defining the environment are executed: {⟨code⟩}\begin{list}{⟨arg one⟩}{⟨arg two⟩}.

`first = {⟨code⟩}` default: *not used*  
 Executes {⟨code⟩} when “starting” the environment. The {⟨code⟩} must be passed between braces, is executed right “after” all *list parameters* are done, after the second argument of list, just before the first occurrence of \item: \begin{list}{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}\item.

- Keep in mind that the code set in this key will affect the entire “body” of the environment and therefore the inner levels of the list and the *keyans* environment. It is recommended to set this key per level.
- In the *enumext\** and *keyans\** environments this key is executed after the *listparindent*, *parsep* and *itemindent* keys within the *minipage* environment in which the “item content” is placed.

`after = {⟨code⟩}` default: *not used*  
 Execute {⟨code⟩} “after” finishing the environment. The {⟨code⟩} must be passed between braces.

## 5.4 Keys for start, series and resume

`start = {⟨integer | integer expression⟩}` default: *1*  
 Sets the *start value* of the numbering on the current level. The {⟨integer expression⟩} must be passed between braces, internally is evaluated and pass to the counter defined by *label* key on the current level, i.e. it is equivalent to enter `start={\dimeval{100*⟨value⟩{chapter}}}` or `start={100*⟨value⟩{chapter}}`.

`start* = {⟨integer | string⟩}` default: *not used*  
 Sets the *start value* of the numbering on the current level. Internally ⟨string⟩ is converted and passed as value to the counter defined by *label* key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

The following ⟨keys⟩ are “only” available for the *enumext\** environment and the “first level” of the *enumext* environment and are ignored if set when nested within each other.

`series = {⟨series name⟩}` default: *not used*  
 Stores the *keys* of the *optional argument* of the “first level” of the environment in which it is executed in {⟨series name⟩} which is used as an argument in the key *resume*. The ⟨keys⟩ stored in {⟨series name⟩} are not cumulative and are overwritten if the same {⟨series name⟩} is used again.

`resume = {⟨series name⟩}` default: *not used*  
 Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` key was executed. If passed *without value* this will only set *start value* continue the numbering from the last environment in which `series={⟨series name⟩}` or `resume={⟨series name⟩}` is not present and if the *save-ans* key is active it will continue the numbering from the last environment in which it was executed. The *start value* can be overwritten using *start* or *start\** keys.

`resume* ⟨value forbidden⟩` default: *not used*  
 Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` or `resume={⟨series name⟩}` keys are NOT present, if the *save-ans* key is active it will continue the numbering from the last environment in which it was executed. The *start value* can be overwritten using *start* or *start\** keys.

- For security reasons the *series* key will never save in {⟨series name⟩} the keys *series*, *resume*, *resume\**, *save-ans*, *save-key*, *start\** and *start*. When using the key `resume={⟨series name⟩}` it will have hierarchy in the ⟨keys⟩ that are saved in {⟨series name⟩}, in order to establish the value of a ⟨key⟩ already saved in {⟨series name⟩} it must be placed to the “right” of `resume={⟨series name⟩}`, the same thing happens with the *resume\** key, the exception is the *save-ans* key that must be placed on the “left” if you want to start the numbering with its value. The *resume* key passed “without value” must be exactly “without value”, i.e. `resume=` cannot be used and if executed before *resume\** it will affect the *start value*.

## 5.5 Keys for multicol

`columns = {⟨integer⟩}` default: *1*  
 Set the *number of columns* to be used by the *multicols* environment within the environment. The value must be a positive integer less than or equal to 10.

`columns-sep = {⟨rigid length⟩}` default: *by level*  
 Set the *space between columns* used by the *multicols* environment within the environment. Internally sets the value of \columnsep, by default its value is equal to the sum of the values set in the keys *labelwidth* and *labelsep* of the current level.

- The \footnote{⟨text⟩} command in the nested levels of *multicols* will not work as expected, prefer the use of \footnotemark[⟨number⟩] inside the environment and \footnotetext[⟨number⟩]{⟨text⟩} outside the environment or via the *after* key.
- By default the *hyperref* package does not provide support for \footnotemark[⟨number⟩] and \footnotetext[⟨number⟩]{⟨text⟩}, but when *tagged* PDF is enabled and the package is loaded the \footnotemark[⟨number⟩] and \footnotetext[⟨number⟩]{⟨text⟩} commands create links correctly.

## 5.6 Keys for minipage

`mini-env = {⟨rigid length⟩}`

default: *not used*

Sets the *width* of the `minipage` environment on the “right side”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “left side”, taking `\linewidth` as the maximum reference value.

`mini-sep = {⟨rigid length⟩}`

default: `0.3333em`

Sets the *space between* the `minipage` environment on the “left side” and the `minipage` environment on the “right side”. This separation is applied together with `\hfill`.

### 5.6.1 The command `\miniright`

---

```
\miniright \begin{enumext}[mini-env=⟨rigid length⟩] ⟨item's before⟩ \item \miniright ⟨content⟩ \end{enumext}
\begin{enumext}[mini-env=⟨rigid length⟩] ⟨item's before⟩ \item \miniright*⟨content⟩ \end{enumext}
```

---

The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”.

The *starred argument* “\*” inhibits the use of `\centering` command i.e. the usual  $\TeX$  justification is maintained in the `minipage` on the “right side”.

• The `\footnote{⟨text⟩}` command in `minipage` environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key (see §1.3.6 for full support).

• By default the `hyperref` package does not provide support for `\footnotemark[⟨number⟩]` and `\footnotetext[⟨number⟩]{⟨text⟩}`, but when *tagged PDF* is enabled and the package is loaded the `\footnotemark[⟨number⟩]` and `\footnotetext[⟨number⟩]{⟨text⟩}` commands create links correctly.

### 5.6.2 The key `mini-right`

In the *horizontal list environments* `enumext*` and `keyans*` it is not possible to use the `\miniright` command and the `mini-right` key must be used instead.

`mini-right = {⟨content⟩}`

default: *not used*

Set the *content* for the drawing or tabular to be placed in the `minipage` environment on the “right side” by starting it with `\centering`. The `{⟨content⟩}` must be passed between braces.

`mini-right* = {⟨content⟩}`

default: *not used*

Same as above, but *without* starting with `\centering`.

## 6 The storage system

The entire mechanism for “*storing content*” it is activated according to `save-ans` key on the “*first level*” of `enumext` or `enumext*` environments and it is ignored if they are established when they are nested inside each other. Only when this `⟨key⟩` is “*active*” the `\anskey` command and the environments `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

<pre>\begin{enumext}[save-ans={⟨store name⟩}]   \item Text \anskey{answer}   \item Text     \begin{keyans}       ...     \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={⟨store name⟩}]   \item Text \anskey{answer}   \item Text     \begin{keyanspic}       ...     \end{keyanspic} \end{enumext}</pre>
---	---

By executing the key `save-ans={⟨store name⟩}` the entire “*structure*” of the environment (excluding the *first level*) including the *optional argument* passed to the inner levels or the environment nested in it, along with the `⟨content⟩` passed to `\anskey` or `anskey*`, the current `⟨labels⟩` for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be “*stored*” in a *sequence* `{⟨store name⟩}` and at the same time will be “*stored*” (without the “*structure*” or *optional argument*) in a *prop list* `{⟨store name⟩}`.

For security reasons the *optional argument* of the inner levels or the nested environment are *filtered* by excluding all `⟨keys⟩` related to the “*storage system*” (§6.1) along with the keys `mini-env`, `mini-sep`, `mini-right`, `mini-right*`, `series`, `resume` and `resume*` when storing in *sequence* `{⟨store name⟩}` set by `save-ans` key.

### 6.1 Keys for storage system

The only `⟨keys⟩` available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the `⟨keys⟩` described in this section must be passed directly in the *optional argument* of the “*first level*” of the environment in which the key `save-ans` is executed. The key `save-ans` should NOT be passed with the command `\setenumext`.

`save-ans = {⟨store name⟩}` default: *not set*  
 Sets the *name* of the *sequence* and *prop list* in which the {⟨contents⟩} will be “stored” by `\anskey` and `\anskey*` in `enumext` and `enumext*` environments and the current {⟨labels⟩} for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic`. If the *sequence* or *prop list* {⟨store name⟩} does not exist, it will be created globally and will not be *overwritten* if the key is used again.

`save-key = {⟨key list⟩}` default: *not set*  
 This key *overrides* the default “stored keys” of the *optional argument* of the inner levels or nested environment that will be passed to the *sequence*. The {⟨key list⟩} passed to this key ignores any {⟨keys⟩} in the “stored structure” and must be passed between braces. For example, if we execute at a second level:

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{enumext}[nosep, columns=2, save-key={columns=3}]
      ...
    \end{enumext}
\end{enumext}
```

The “stored keys” by default in the *sequence* {⟨store name⟩} would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite and the “stored key” in the *sequence* {⟨store name⟩} are only `columns=3` ignoring all the others.

`save-sep = {⟨text symbol⟩}` default: {, }  
 Sets the *text symbol* that will separate the current {⟨label⟩} to the *optional argument* passed to the `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` and storing them in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. The {⟨text symbol⟩} must always be passed between braces, whitespace ‘`\`’ is preserved within the braces and only affects the “stored content” and not what is displayed when using the `show-ans` or `show-pos` keys.

#### 6.1.1 Keys for label and ref

`save-ref = {⟨true | false⟩}` default: *false*  
 Activates the “internal label and ref” mechanism for referencing “stored content” in *prop list* {⟨store name⟩} set by `save-ans` key. To reference the location of the “stored content” within the environment you must use `\ref{⟨store name : position⟩}`, where {⟨position⟩} corresponds to the position occupied by the “stored content” in the *prop list* {⟨store name⟩} returned by the `show-pos` key. For example `\ref{test:4}` will return `3`. (b) which corresponds to the location of the “stored content” at position `4` in *prop list* `test` within the environment in which the key `save-ans=test` was set.

`mark-ref = {⟨symbol⟩}` default: `\%`  
 Sets the *symbol* that will be displayed by the `\printkeyans` command only if the `hyperref` package is detected and the `save-ref` key are active. This “symbol” is used as a “link” between the environment in which the `save-ans` key was used and the place where the command is executed.

#### 6.1.2 Keys for wrap and display

`wrap-ans = {⟨code {#1} more code⟩}` default: `\fbox+\parbox{#1}`  
 Wraps the *argument* passed to the `\anskey` and the *body* in `\anskey*` environment referenced by {#1} when using the `show-ans` or `show-pos` keys. The {⟨code⟩} must be passed between braces and only affects the *argument* or *body* and NOT the “stored content” in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.

`wrap-opt = {⟨code {#1} more code⟩}` default: `[{#1}]`  
 Wraps the *optional argument* passed to the `\item*` and `\anspic*` referenced by {#1} in the `keyans`, `keyans*` and `keyanspic` environments when using the `show-ans` or `show-pos` keys. The {⟨code⟩} must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.

`show-ans = {⟨true | false⟩}` default: *false*  
 Displays the *argument* passed to the `\anskey`, the *body* for `\anskey*` environment, the {⟨label⟩} for `\item*` and `\anspic*` at the place where it is executed. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

`mark-ans = {⟨symbol⟩}` default: `\textasteriskcentered`  
 Sets the *symbol* to be displayed in the left margin for `\anskey`, `\anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

`mark-pos = {⟨left | right⟩}` default: *left*  
 Sets the *aligned* of the symbol defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.



6.1.3 Keys for debug and checking

- show-pos = {<true | false>}

default: false

Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* {<store name>} set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.
- check-ans = {<true | false>}

default: false

Enables the *checking answer* mechanism displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “only one answer” or “only one execution” of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.
- no-store <value forbidden>

default: not used

This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* {<store name>} set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “without” using the `\anskey`, “without” use `anskey*`, “without” interfering with the `check-ans` key and “without” storing an unwanted structure in the *sequence* {<store name>}.

6.2 The command \anskey

`\anskey` `\anskey` [<keys>] {<content>}

The command `\anskey` takes a mandatory non empty argument {<content>} and “stores” it in the *sequence* and *prop list* {<store name>} set by `save-ans` key. By design the command cannot be nested or passed *verbatim material* in the argument and it is assumed that each *numbered* `\item` or `\item*` within the environment in which it is active it has a “single execution” of `\anskey` unless `\item` or `\item*` open a nested level or use the `no-store` key.

If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by `LaTeX` will be used.

The `\anskey` command is available for all levels of the `enumext` environment and the `enumext*` environment, but is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

6.2.1 Keys for \anskey

By default the {<content>} passed to `\anskey` when “storing” in the *sequence* {<store name>} has the form `\item` <content>, the following <keys> allow modifying the way in which it is “stored” in the *sequence*.

- break-col <value forbidden>

default: not used

Stores {<content>} in the *sequence* {<store name>} of the form `\columnbreak` `\item` <content>.
- item-join = {<columns>}

default: not set

Set the *number of columns* to be used for `\item`(<columns>) and stores {<content>} in the *sequence* {<store name>} of the form `\item`(<columns>) <content>.
- item-star <value forbidden>

default: not used

Stores {<content>} in the *sequence* {<store name>} of the form `\item*` <content>.
- item-sym\* = {<symbol>}

default: not set

Sets the *symbol* for `\item*` when using the key `item-star` and stores {<content>} in the *sequence* {<store name>} of the form `\item*` [<symbol>] <content>. The *symbol* can be in text or math mode, for example `item-sym*={$\ast$}` stores `\item*[$\ast$]` <content>.
- item-pos\* = {<rigid length>}

default: not set

Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores {<content>} in the *sequence* {<store name>} of the form `\item*` [<symbol>] [<offset>] <content>.

Example

```
\begin{enumext}[save-ans=test,show-ans=true]
  \item* Text containing our instructions or questions. \anskey{<first answer>}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.\anskey{<second answer>}
    \end{enumext}
  \item Text containing our instructions or questions. \anskey{<third answer>}
  \item Text containing our instructions or questions. \anskey{<fourth answer>}
\end{enumext}
```

1. Text containing our instructions or questions.

\* first answer
2. Text containing our instructions or questions.

(a) Question.

\* second answer
3. Text containing our instructions or questions.

\* third answer
4. Text containing our instructions or questions.

\* fourth answer

6.3 The environment anskey\*

`anskey*` `\begin{anskey*}[\langle key = val \rangle] \langle body content \rangle \end{anskey*}`

The environment `anskey*` takes a mandatory `\langle body content \rangle` and “stores” it in the *sequence* and *prop list* `\{store name\}` set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperLink` and `\hypertarget` will be used, otherwise the usual “*label and ref*” system provided by  $\LaTeX$  will be used.

By design the environment cannot be nested but full supports “*verbatim material*” in the body and it is assumed that each numbered `\item` or `\item*` within the environment in which it is active it has a “*single execution*” unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the `scontents` package, for the correct operation `\begin{anskey*}` and `\end{anskey*}` must be in different lines, all `\langle keys \rangle` must be passed separated by commas and “without separation” of the start of the environment. Comments “%” or “any character” after `\begin{anskey*}` or `[\langle key = val \rangle]` on the same line are NOT supported, the package `scontents` will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{anskey*}` on the same line the package `scontents` will return a “warning” message.

6.3.1 Keys for anskey\*

The `anskey*` environment uses the same `\langle keys \rangle` as the `\anskey` command next to the keys inherited from package `scontents`. The environment is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

`write-env` = `\{ \langle file.ext \rangle \}` default: *not used*

Sets the name of the `\langle external file \rangle` in which the `\langle contents \rangle` of the environment will be written. The `\langle file.ext \rangle` will be created in the working directory, relative or absolute paths are not supported. If `\langle file.ext \rangle` does not exist, it will be created or overwritten if the `overwrite` key is used.

`overwrite` = `\{ \langle true | false \rangle \}` default: *false*

Sets whether the `\langle file.ext \rangle` generated by `write-env` from the `anskey*` environment will be rewritten.

`force-eol` = `\{ \langle true | false \rangle \}` default: *false*

Sets if the *end of line* for the `\langle stored content \rangle` is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the `fancyvrb` package as `\end{Verbatim}` or another environment that does not support a comments “%” after closing `\end{Verbatim}%`.

- For security reasons the keys `store-env`, `print-env` and `write-out` they have been left disabled. It is recommended that you review the `scontents`[4] documentation to understand how the keys described here work.

Example

```
\begin{enumext}[save-ans=test,show-pos=true,start=5]
  \item* Text containing our instructions or questions.
    \begin{anskey*}[item-star]
      \langle first answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          \langle second answer \rangle
        \end{anskey*}
      \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle third answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle fourth answer \rangle
    \end{anskey*}
\end{enumext}
```

- |   |   |
|---|---|
| * 5. Text containing our instructions or questions. | 7. Text containing our instructions or questions. |
| [5] <div>First answer with verbatim</div>           | [7] <div>third answer</div>                       |
| 6. Text containing our instructions or questions.   | 8. Text containing our instructions or questions. |
| (a) Question.                                       | [8] <div>fourth answer</div>                      |
| [6] <div>second answer</div>                        |   |



6.4 The environments keyans and keyans\*

keyans

keyans\*

```
\begin{keyans}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}
\begin{keyans*}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans*}
```

The `keyans` and `keyans*` environments are “*enumerated list*” environments designed for “*multiple choice*” questions activated by the `save-ans` key. This environments can NOT be nested and must always be at the “*first level*” of the `enumext` environment, the commands `\item` and `\item[\langle custom \rangle]` work in the usual and the command `\item(\langle columns \rangle)` is available for the `keyans*` environment.

```
\begin{enumext}[save-ans=test]
  \item \langle item content \rangle
  \begin{keyans}[\langle key = val \rangle]
    \item \langle item content \rangle
    \item [\langle custom \rangle] \langle item content \rangle
    \item* \langle item content \rangle
    \item*[\langle content \rangle] \langle item content \rangle
  \end{keyans}
\end{enumext}

\begin{enumext}[save-ans=test]
  \item \langle item content \rangle
  \begin{keyans*}[\langle key = val \rangle]
    \item \langle item content \rangle
    \item [\langle custom \rangle] \langle item content \rangle
    \item* \langle item content \rangle
    \item*[\langle content \rangle] \langle item content \rangle
  \end{keyans*}
\end{enumext}
```

The `\langle keys \rangle` set in the *optional argument* of the environment are the same (almost) as those of the `enumext` and `enumext*` environments and have *higher precedence* than those set by `\setenumext[\langle keyans \rangle]{\langle key = val \rangle}` or `\setenumext[\langle keyans* \rangle]{\langle key = val \rangle}`. If the *optional argument* is not passed or the `\langle keys \rangle` are not set by `\setenumext`, the default values will be the same as the “*second level*” of the `enumext` environment with the difference in the `\langle label \rangle` which will be set to `label=\Alph*`.

6.4.1 The \item\* in keyans and keyans\*

\item\*

```
\item*
\item*[\langle content \rangle]
```

The `\item*` and `\item*[\langle content \rangle]` command “*store*” the current `\langle label \rangle` set by `label` key next to the *optional argument* `\langle content \rangle` in *sequence* and *prop list* `{\langle store name \rangle}` set by `save-ans` key in the “*first level*” of the `enumext` or `enumext*` environments.

The *starred argument* “`*`” cannot be separated by spaces “`␣`” from the command, i.e. `\item*` and the *optional argument* does “*NOT*” support *verbatim content*. By design it is assumed that the `\item*` will only appear “*once*” within the environment.

🔗 The behavior of `\item*` in `keyans` and `keyans*` environments is NOT the same as in the `enumext` or `enumext*` environments.

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.

  \begin{keyans*}[nosep,columns=2]
    \item Choice
    \item* Correct choice
    \item Choice
    \item Choice
    \item Choice
  \end{keyans*}

  \item Text containing a question and image.

  \begin{keyans}[nosep,mini-env={0.4\linewidth}]
    \item Choice
    \item Choice
    \item Choice
    \item Choice
    \item*[\langle note \rangle] Correct choice
    \miniright
    \includegraphics[scale=0.25]{example-image-a}
    Some text
  \end{keyans}
\end{enumext}
```

1. Text containing a question.

2. Text containing a question and image.

A) Choice

C) Choice

E) Choice

\* B) Correct choice

D) Choice


A) Choice

B) Choice

C) Choice

D) Choice

\* E) [note] Correct choice



Some text

## 6.5 The environment keyanspic

---

```
keyanspic \begin{keyanspic}[\langle key = val \rangle] \anspic*[\langle content \rangle]{\langle drawing or tabular \rangle} \end{keyanspic}
```

---

The `keyanspic` environment is an “*enumerated list*” environment activated by the `save-ans` key that has the same configuration for “*spacing*” and `\label` as the `keyans` environment that uses the `\anspic` command instead of `\item`. It is intended for placing *drawings or tabular* with `\label` centered *above* or *below* in a *single line* or *upper and lower* layout style. A representation of the output can be seen in the figure 6.

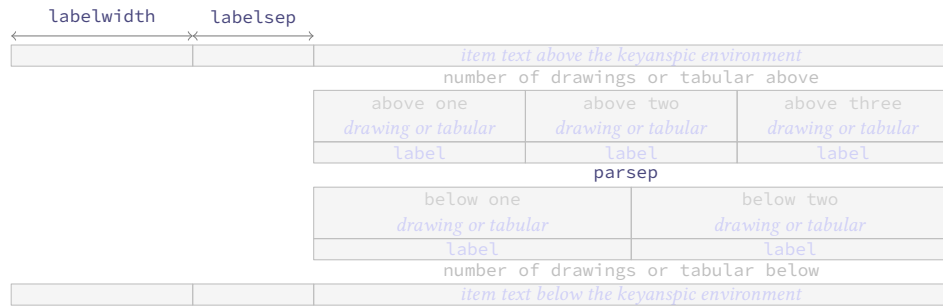


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

When the `keyanspic` environment is used *without keys* the `\label`s are centered *below* the *drawings or tabular* in a *single line* layout style.

This environment cannot be nested and must always be at the “*first level*” of the `enumext` environment, the `\item` command is disabled and keys cannot be set using `\setenumext`.

### 6.5.1 Keys for keyanspic

`label-pos = {\langle above | below \rangle}` default: *below*

Set the *position* of `\label` to be centered “above” or “below” *drawings or tabular* when the `\anspic` command is executed.

`label-sep = {\langle rubber length | rigid length \rangle}` default: *internal adjustment*

Set the *vertical spacing* between the `\label` centered “above” or “below” and *drawings or tabular* when running the `\anspic` command.

`layout-sty = {\langle n° upper , n° lower \rangle}` default: *not set*

Set the *number of drawings or tabular* that will be distributed “upper” and “lower” within the environment when executing the `\anspic` command. The value must be passed in braces and if not set or the `\langle n° lower \rangle` is omitted the *drawings or tabular* will be put on a *single line*.

`layout-sep = {\langle rubber length | rigid length \rangle}` default: *adjusted parsep from keyans*

Set the *vertical separation* between the number of *drawings or tabular* placed at the “upper” and “lower” within the environment when executing the `\anspic` command. Internally adjusts the `parsep` value taken from the `keyans` environment.

`layout-top = {\langle rubber length | rigid length \rangle}` default: *adjusted topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. Internally adjust the value of `topsep` taken from `keyans` environment.

### 6.5.2 The command \anspic

---

```
\anspic \anspic{\langle drawing or tabular \rangle}
\anspic*[\langle content \rangle]{\langle drawing or tabular \rangle}
```

---

The `\anspic` command take three arguments, the *starred argument* ‘`*`’ store the current `\label` next to the *optional argument* `\langle content \rangle` in *sequence* and *prop list* `{\langle store name \rangle}` set by `save-ans` key.

The *starred argument* ‘`*`’ cannot be separated by spaces ‘`␣`’ from the command, i.e. `\anspic*` and the *optional argument* does “NOT” support *verbatim content*. By design it is assumed that the *starred argument* ‘`*`’ will only appear “once” within the environment.

#### Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
\item Question with images and labels below.

\begin{keyanspic}[layout-sty={3,2}]
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}
```


```
\item Question with images and labels above.


\begin{keyanspic}[label-pos=above, layout-sty={3,2}]
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}


\item Question with images and labels below on a single line.


\begin{keyanspic}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}
\end{enumext}
```


1. Question with images and labels below.

  
A)

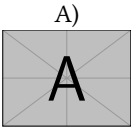
  
B)

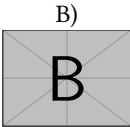
  
C)

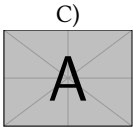
  
D)


  
\* E)[note]


2. Question with images and labels above.

  
A)


  
B)

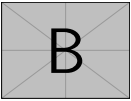
  
C)


  
D)


  
\* E)[note]


3. Question with images and labels below on a single line.

  
A)

  
B)

  
C)

  
D)

  
\* E)[note]

💡 Preferably use `label-pos=above` when creating a *tagged* PDF, this will preserve the reading order and navigation of the document.

6.6 Printing stored content

6.6.1 The command `\getkeyans`

```
\getkeyans \getkeyans{<store name> : <position>}
```

The command `\getkeyans` prints the “stored content” in *prop list* `{<store name>}` defined by `save-ans` key in the `<position>` returned by the `show-pos` key. The “stored content” can only be accessed *after* it is stored, if `{<store name>}` does not exist the command will return an error.

The form taken by the argument `{<store name> : <position>}` is the same as that used to generate the “internal label and ref” system when `save-ref` key are active, so to refer to a “stored content”. For example `\getkeyans{test:4}` will return the “stored content” at position 4 of the environment in which the key `save-ans=test` was set.

6.6.2 The command `\foreachkeyans`

```
\foreachkeyans \foreachkeyans[<key = val>]{<store name>}
```

The command `\foreachkeyans` goes through and executes the command `\getkeyans` on the contents in *prop list* `{<store name>}`. If you pass without options run `\getkeyans` on all contents in *prop list* `{<store name>}`.

### Options for command

<code>sep = {&lt;code&gt;}</code>	default: <code>;</code>
Establishes the <i>separation</i> between “each” <code>{&lt;content&gt;}</code> stored in <i>prop list</i> <code>{&lt;store name&gt;}</code> . For example, you can use <code>sep={\[\[10pt]}</code> for vertical separation of stored contents.	
<code>step = {&lt;integer&gt;}</code>	default: <code>1</code>
Sets the <i>step</i> (increment) applied to the value set by key <code>start</code> for each <code>{&lt;content&gt;}</code> stored in <i>prop list</i> <code>{&lt;store name&gt;}</code> . The value must be a <i>positive integer</i> .	
<code>start = {&lt;integer&gt;}</code>	default: <code>1</code>
Sets the <i>position</i> of the <i>prop list</i> <code>{&lt;store name&gt;}</code> from which execution will start. The value must be a <i>positive integer</i> .	
<code>stop = {&lt;integer&gt;}</code>	default: <code>0</code>
Sets the <i>position</i> of the <i>prop list</i> <code>{&lt;store name&gt;}</code> from which execution it will finish executing. The value must be a <i>positive integer</i> .	
<code>before = {&lt;code&gt;}</code>	default: <i>empty</i>
Sets the <code>{&lt;code&gt;}</code> that will be executed <i>before</i> each <code>{&lt;content&gt;}</code> stored in <i>prop list</i> <code>{&lt;store name&gt;}</code> . The <code>{&lt;code&gt;}</code> must be passed between braces.	
<code>after = {&lt;code&gt;}</code>	default: <i>empty</i>
Sets the <code>{&lt;code&gt;}</code> that will be executed <i>after</i> each <code>{&lt;content&gt;}</code> stored in <i>prop list</i> <code>{&lt;store name&gt;}</code> . The <code>{&lt;code&gt;}</code> must be passed between braces.	
<code>wrapper = {&lt;code&gt;{#1} more code}</code>	default: <i>empty</i>
Wraps the <code>{&lt;content&gt;}</code> stored in <i>prop list</i> <code>{&lt;store name&gt;}</code> referenced by <code>{#1}</code> . The <code>{&lt;code&gt;}</code> must be passed between braces. For example <code>\foreachkeys[wrapper={\makebox[1em][l]{#1}}]{&lt;store name&gt;}</code> .	

#### 6.6.3 The command `\printkeys`

---

```
\printkeys \printkeys{<store name>}
\printkeys[<keys>]{<store name>}
\printkeys*[<keys>]{<store name>}
```

---

The command `\printkeys` prints “all stored content” in *sequence* `{<store name>}` defined by `save-ans` key placing this inside the `enumext` environment by default or the `enumext*` environment if the *starred argument* ‘\*’ is used.

The “*stored content*” can only be accessed *after* it is stored in the *sequence*, if `{<store name>}` does not exist the command will return an error.

The *optional argument* allows managing the `<keys>` in the “*first level*” of the environment in which the “*stored content*” of the *sequence* `{<store name>}` will be printed, if the *starred argument* ‘\*’ is used it will be `enumext*` otherwise `enumext`.

The default values for the “*first level*” are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep`, `first=\small`, `font=\small` and `columns=2`. For the inner levels of the environment `enumext` saved in the *sequence* `{<store name>}` the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the *sequence* `{<store name>}` it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

Since the command encapsulates by default the `enumext` environment or the `enumext*` environment, we must take some considerations:

- If we execute `\printkeys*{<store name>}` and the *sequence* `{<store name>}` already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeys*{<store name>}` and the *sequence* `{<store name>}` contains any `enumext` environments, they will start with the `<keys>` set for the first level unless they are set in the *optional argument* or `save-key` is used to modify it.
- If we execute `\printkeys{<store name>}` and the *sequence* `{<store name>}` contains any environment `enumext*`, they will start with the `<keys>` set by default unless they are set in the *optional argument* or `save-key` is used to modify it.

The default values for the “*first level*” of `\printkeys` commands and `\printkeys*` are established using `\setenumext[<print>]{<keys>}` and `\setenumext[<print*>]{<keys>}`.

If we need to set the `<keys>` for the environment `enumext` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print>]{<keys>}` and if we need to set the `<keys>` for the environment `enumext*` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print*>]{<keys>}`.

Example

```
\begin{enumext}[save-ans=sample,columns=1,show-pos=true,nosep,save-ref=true]
  \item Factor  $3x+3y+3z$ . \anskey{$3(x+y+z)$}
  \item True False

  \begin{enumext}[nosep]
    \item \LaTeXe\ is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

  \begin{enumext}[nosep]
    \item You use linux? \anskey{Yes}
    \item Rate the following package and class
      \begin{enumext}[nosep]
        \item \texttt{xsim} \anskey{very good}
        \item \texttt{exsheets} \anskey{obsolete}
      \end{enumext}
    \end{enumext}
\end{enumext}

The answer to \ref{sample:4} is \getkeyans{sample:4} and the answers to
all the worksheets are as follows:

\printkeyans{sample}
```

1. Factor  $3x + 3y + 3z$ .

[1]

2. True False

(a)

[2]

3. Related to Linux

(a)

[3]

(b) Rate the following package and class

i.

[4]

ii.

[5]

The answer to 3.(b).i is very good and the answers to all the worksheets are as follows:

1.  $3(x + y + z)$  ✖
2. (a) Very True! ✖
3. (a) Yes ✖
- (b) i. very good ✖
- ii. obsolete ✖

7 Full examples

Here I will leave as an example some adaptations questions taken from TeX-SX. The examples are attached to this documentation and can be extracted from your PDF viewer or from the command line by running:

```
$ pdftdetach -saveall enumext.pdf
```

and then you can use the excellent [arara](#)<sup>1</sup> tool to compile them.

Example 1

Adapted from the response given by Enrico Gregorio in [Squares for answer choice options and perfect alignment to mathematical answers](#) .

1. La velocità di  $1,00 \times 10^2$  m/s espressa in km/h è:  $10^{-15}$  m). Qual è la relazione tra queste due unità di misura?
- A

36 km/h.

B

360 km/h.
- C
- 27,8 km/h.

D

A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm.}$ 

B

C

D

2. In fisica nucleare si usa l'angstrom (simbolo:  $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$ ) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15} \text{ m}$ )3. La velocità di  $1,00 \times 10^2$  m/s espressa in km/h è:

<sup>1</sup>The cool TeX automation tool: <https://www.ctan.org/pkg/arara>

©2024 by Pablo González L

19 / 157

- A

36 km/h.

B

360 km/h.

C

27,8 km/h.

D

 $3,60 \times 10^8$  km/h.
4. In fisica nucleare si usa l'angstrom (simbolo:  $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$ ) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15} \text{ m}$ ). Qual è la relazione tra queste due unità di misura?
- A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$ .
- B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$ .
- C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$ .
- D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$ .


1. B

2. A

3. B

4. A

Example 2

Adapted from the response given by Florent Rougon in [Multiple choice questions with proposed answers in random order — addition of automatic correction \(cross mark\)](#) 

1. La velocità di  $1,00 \times 10^2 \text{ m/s}$  espressa in km/h è:

A

36 km/h.

B

360 km/h.

C

27,8 km/h.

D

 $3,60 \times 10^8$  km/h.
2. In fisica nucleare si usa l'angstrom (simbolo:  $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$ ) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15} \text{ m}$ ). Qual è la relazione tra queste due unità di misura?
- A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$ .
- B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$ .
- C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$ .
- D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$ .

1. B

2. A

3. La velocità di  $1,00 \times 10^2 \text{ m/s}$  espressa in km/h è:

A

36 km/h.

B

360 km/h.

C

27,8 km/h.

D

 $3,60 \times 10^8$  km/h.

4. In fisica nucleare si usa l'angstrom (simbolo:  $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$ ) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15} \text{ m}$ ). Qual è la relazione tra queste due unità di misura?

A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$ .

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$ .

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$ .


D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$ .

3. A

4. A

Example 3

A “simple multiple choice” test 

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

I.  $2\alpha + 2\delta = 90^\circ$

II.  $\alpha = \delta$

III.  $\angle EDF = 45^\circ$

A

I only

B

II only

C

I and II only

D

I and III only

E

I, II, and III

3. Third type of questions

(1)  $2\alpha + 2\delta = 90^\circ$

(2)  $\angle EDF = 45^\circ$

A

value

B

value

C

value

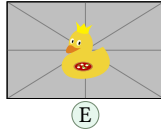
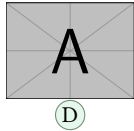
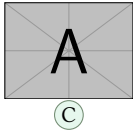
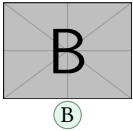
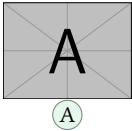
D

value

E

value

4. Question with image and label below:



5. Question with image on left side:
- A

value
- B

value
- C

value
- D

correct
- E

value



Test keys

1. B),  $x = 5$


2. D


3. C, some note
4. E, A duck


5. D, other note




Example 4

A “simple worksheet” using ducks :) 


 Factor  $x^2 - 2x + 1$

 Factor  $3x + 3y + 3z$

The following questions need to be cuaqtified :)

 True False

- (a)  $\alpha > \delta$
- (b)  $\LaTeX$  is cool?

 Related to Linux

- (a) You use linux?
- (b) Usually uses the package manager?
- (c) Rate the following package and class
  - i. `xsim-exam`
  - ii. `xsim`
  - iii. `exsheets`

The answer to 1 is  $(x - 1)^2$  and the answer to 3.(a) is False.

- |                   |   |                                 |   |
|-------------------|---|---------------------------------|---|
| 1. $(x - 1)^2$    | ✖ | (b) Yes, dnf                    | ✖ |
| 2. $3(x + y + z)$ | ✖ | (c) i. doesn't exist for now :( | ✖ |
| 3. (a) False      | ✖ | ii. very good                   | ✖ |
| (b) Very True!    | ✖ | iii. obsolete                   | ✖ |
| 4. (a) Yes        | ✖ |                                 |   |

Example 5

Adapted from the response given by Stephen in SAT like question format 





<div>1</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>	<div>3</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>
<div>2</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>	<div>4</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>







1. A)                      2. C)                      3. B)                      4. D)

8 Tagged PDF examples

This section is just to show the compatibility of enumext with tagged PDF using lua<sup>l</sup>atex. The attached files here are just for testing and are intended as examples and, in a way, to simplify the time of Matthew Bertucci (@mbertucci) when he sees this excellent package and adds it to The LaTeX Tagged PDF repository.

To compile the tests with lua<sup>l</sup>atex-dev the packages multicol, scontents, unicode-math, geometry, graphicx and hyperref are required.

- The file enumext-01.tex contains the basic tests for the enumext and enumet\* environments and the nesting between them plus the use of the label, labelwidth, labelsep, ref, align and wrap-label keys. Source file  and tagged PDF .
- The file enumext-02.tex contains the tests for the enumext and enumet\* environments and the support for minipage and multicols environments using the keys columns, columns-sep, mini-env, mini-right and \miniright command. Source file  and tagged PDF .

- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file  and *tagged PDF* .
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file  and *tagged PDF* .
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file  and *tagged PDF* .

## 9 The way of non-enumerated lists

It is possible to use (or abuse) the `enumext` environment to mimic *non-enumerated* list environments such as `itemize` and `description`, clearly the `\keys` to “store answers”, the `keyans` and `keyanspic` environments lose their sense and it is not the focus of the main of this package, but, why not to do it?.

Here I leave as an example other uses of the `enumext` environment that can be helpful for specific purposes. The “trick” to generate these *fake environments* is set `label={}` or `label={\some}` and play with the `list-indent`, `list-offset`, `font` and `wrap-label` keys.

### Fake itemize environment

Here we set the `label` key using the default settings in  $\TeX$  for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

- |                     |                     |
|---------------------|---------------------|
| • First level item  | * First level item  |
| – Second level item | ◇ Second level item |
| * Third level item  | ○ Third level item  |
| · Fourth level item | ★ Fourth level item |
| • First level item  | * First level item  |

### Fake description environment

Here we set `label={}` and `list-indent=2.5em`, `font=\bfseries`.

**Something** A short one-line description.

This is an entry *without* a label.

**Something** A short *one-line* description text.

**Something long** A much *longer* description text may take more than one line or more than one paragraph.

  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

If we add `list-indent=0pt` you get *widest style*:

**Something** A short one-line description.

This is an entry *without* a label.

**Something** A short *one-line* description text.

**Something long** A much *longer* description text may take more than one line or more than one paragraph.

  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- The small space at the beginning of the “unlabeled entry” corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.

### Description indented by label

Here we set `label={}` and we will give a convenient value to `labelsep` and `labelwidth`, for example we can take as reference our *longest label* and pass it as value using:

```
\newlength{\descitemwd}
\settowidth{\descitemwd}{\textbf{Something long}}
```

and then use `labelsep=4pt`, `labelwidth=\descitemwd`, `font=\bfseries`.

**Something** A short one-line description.

This is an entry *without* a label.

**Something** A short one-line description.

**Something long** A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

The environment can be translated so that the `\labels` are on the left margin calculating the value passed to the `list-offset` key, in this case it will be equal to the sum of the values set by the `labelwidth` and `labelsep` keys finally resulting as `list-offset={-\descitemwd - 4pt}`.

SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. If we add <code>align=right</code> it will look like this:
SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

At this point we have used `list-offset={-\descitemwd - 4pt}` instead of `list-offset={-\labelwidth - \labelsep}`, this is because the parameters `\labelwidth` and `\labelsep` take the default values, as if we had not set `label`.

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label` and `wrap-label*` keys comes into play. Unlike the `enumitem` package, the `align` key only supports three options, so what we will do is create a command in the style `\parleft` of `enumitem` that allows us to place *multiline labels* using `\parbox`.

```
\NewDocumentCommand \labelbx { s +m }
{
  \%
  \IfBooleanTF{#1}
  {
    \strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}%
    \strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}%
  }
}
```

Now we just need to set `wrap-label*={\labelbx{#1}}`.

SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
SoMeThInG	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[5]` package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in [Create a fake label ref using list](#) and the answer given by David Carlisle in [Change the use of label ref by data save in an array \(list\)](#) I managed to create a more solid code than the original version, now using the `l3prop[11]` and `l3seq[11]` modules together with the `hyperref[8]` and `enumitem[6]` packages, which did the job, but with some limitations.

As time went by I took these limitations as a personal challenge which I called “*reinventing the wheel*”, since there were packages and classes that did more or less what I was looking for, but did not fit my simple requirements. This “*reinventing the wheel*” finally ended up becoming `enumext`.

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn’t have in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.  
Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

The “*random*” type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with  $\text{\LaTeX}$ , that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

## Why has it taken so long?

One of the setbacks, beyond my laziness, was including compatibility with *tagged* PDF. To be honest, it's something I never considered at any point, but I firmly believe that being able to create *accessible documents* provides a great opportunity in the world of mathematics education. From my perspective as a *high school* teacher, beyond theorems and deep mathematics, the use of exercise lists is one of the most common things. Being able to open the way to work in parallel with those who have different abilities is really important and I regret not having looked into this in the past. I hope that `enumext` serves this purpose and inspires more users and authors to follow this path.

## 10 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores  $\LaTeX$  contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2024.
- [5] The  $\LaTeX$  Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ $\LaTeX 2_{\epsilon}$ : An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The  $\LaTeX$  Project. “Extensive support for hypertext in  $\LaTeX$ ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The  $\LaTeX$  Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The  $\LaTeX$  Project. “The  $\LaTeX 3$  Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The  $\LaTeX$  Project. “The  $\LaTeX 2_{\epsilon}$  sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.
- [13] The  $\LaTeX$  Project. “ $\LaTeX$  for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2024.
- [14] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [15] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [16] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

## 11 Change history

**v1.0** 2024-10-10 – First public release.

12 Index of Documentation

The italic numbers denote the pages where the corresponding entry is described.

C		F	
Document class:		\footnote	5
article	2	I	
book	2	\itemsep	8
exam	2	K	
letter	2	Keys for \anskey provide by enumext:	
report	2	break-col	13
\columnbreak	4, 13	item-join	13
\columnsep	10	item-pos*	13
Commands provide by enumext:		item-star	13
\anskey	11-14	item-sym*	13
\anspic	11-13, 16	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	17	after	18
\getkeyans	13, 17	before	18
\item*	5-7, 11-13, 15	sep	18
\item	5-7, 10, 11, 13, 15, 16	start	18
\miniright	11	step	18
\printkeyans	6, 12, 18	stop	18
\setenumextmeta	6	wrapper	18
\setenumext	5-7, 11, 12, 15, 18	Keys for anskey* provide by enumext:	
Counters defined by enumext:		break-col	13
enumXiii	4	force-eol	14
enumXii	4	item-join	13
enumXiv	4	item-pos*	13
enumXi	4	item-star	13
enumXviii	4	item-sym*	13
enumXvii	4	overwrite	14
enumXvi	4	write-env	14
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	8, 9
anskey*	11-14, 22	after	10, 11
enumet*	21	align	7, 21, 23
enumext*	4-15, 18	base-fix	8
enumext	4-16, 18, 21, 22	before*	9, 10
kenyans*	9	before	9
keyans*	4-15, 22	below*	9
keyanspic	4, 7, 8, 11-14, 16, 22	below	9
keyans	4-16, 22	check-ans	13
Environments:		columns-sep	4, 10, 21
Verbatim	14	columns	4, 9, 10, 21
center	5	first	10
description	5, 22	font	7
enumerate	1, 3, 5, 23	item-pos*	5, 6
figure	5	item-sym*	5, 6
flushleft	5	itemindent	9, 10
flushright	5	itemsep	8
itemize	5, 22	label-indent	9
list	3, 5, 9, 23	label-pos	16, 17
minipage	3-5, 8-11, 21, 23	label-sep	16
multicols	3, 4, 10, 21	labelsep	3-7, 9, 10, 12, 21, 22
quotation	5	labelwidth	3, 4, 6, 7, 9, 10, 12, 21, 22
quote	5	labelwith	5
tabbing	5	label	7, 8, 10, 15, 21-23
table	5	labewdith	9
task	5	layout-sep	16
trivlist	5	layout-sty	16
verbatim	5	layout-top	16
verse	5	list-indent	3, 9

list-offset .....	3, 9, 22, 23	Labels provide by enumext:	
listparindent .....	9, 10	\Alph* .....	7, 8, 15
mark-ans .....	12	\Roman* .....	7, 8
mark-pos .....	12	\alph* .....	7, 8
mark-ref .....	12	\arabic* .....	7, 8
mini-env .....	4, 9, 11, 21	\roman* .....	7, 8
mini-right* .....	7, 11	\labelsep .....	3, 7
mini-right .....	7, 11, 21	\labelwidth .....	3, 7
mini-sep .....	4, 11	\linewidth .....	11
mode-box .....	7	\listparindent .....	9
no-store .....	11, 13, 14, 22		
noitemsep .....	8		
nosep .....	8, 22	<b>P</b>	
overwrite .....	14	Packages:	
parsep .....	8, 10, 16	enumerate .....	23
partopsep .....	8	enumext .....	1-5, 7, 16, 21, 23, 24
ref .....	4, 8, 21	enumitem .....	3-5, 23
resume* .....	7, 10, 11	fancyvrb .....	14
resume .....	7, 10, 11	footnotehyper .....	5
rightmargin .....	9	geometry .....	21
save-ans .....	4, 6, 10-18, 22	graphicx .....	21
save-key .....	10-12, 18	hyperref .....	4, 5, 10-14, 21, 23
save-ref .....	4, 7, 12-14, 17, 22	l3keys .....	7
save-sep .....	12, 22	l3prop .....	1, 23
series .....	7, 10, 11	l3seq .....	1, 23
show-ans .....	12, 22	multicol .....	1, 2, 4, 21, 23
show-length .....	8	scontents .....	1, 2, 14, 21
show-pos .....	12, 13, 17	task .....	5, 6
start* .....	10	unicode-math .....	21
start .....	10	xsim .....	2
topsep .....	8, 9, 16	\parsep .....	8
widest .....	7	\partopsep .....	8
wrap-ans .....	12		
wrap-label* .....	7, 23	<b>R</b>	
wrap-label .....	7, 21, 23	\raggedcolumns .....	4
wrap-opt .....	12	\ref .....	4
write-env .....	14	\rightmargin .....	9
<b>L</b>		<b>T</b>	
\label .....	4	\topsep .....	8



## 13 Implementation

The most recent publicly released version of `enumext` is available at CTAN: <https://www.ctan.org/pkg/enumext>. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: <https://github.com/pablgonz/enumext/issues>.

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a TeXpert are superfluous, but, after so many years developing this project is the only way to remember what does what.

### 13.1 General conventions

Variables containing `i`, `ii`, `iii` and `iv` are associated by level with the `enumext` environment, variables containing `v` are associated with the `keyans` environment, variables containing `vi` are associated with the `keyanspic` environment, variables containing `vii` are associated with the `enumext*` environment and variables containing `viii` are associated with the `keyans*` environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital “X”.

The temporary function `\__enumext_tmp:n` is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

### 13.2 Initial set up

Start the DocStrip guards.

```
1 <{*package>
```

Identify the internal prefix (L<sup>A</sup>T<sub>E</sub>X3 DocStrip convention) for l3doc class.

```
2 <@@=enumext>
```

### 13.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L<sup>A</sup>T<sub>E</sub>X to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2024-06-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-10-10}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the `multicol` and `scontents` packages are loaded, if not we load it.

```
9 \hook_gput_code:nnn {begindocument} {enumext}
10 {
11   \IfPackageLoadedTF { multicol }
12   {
13     \msg_info:nnn { enumext } { package-load } { multicol }
14   }
15   {
16     \msg_info:nnn { enumext } { package-not-load } { multicol }
17     \RequirePackage{multicol}[2024-05-23]
18   }
19   \IfPackageLoadedTF { scontents }
20   {
21     \msg_info:nnn { enumext } { package-load } { scontents }
22   }
23   {
24     \msg_info:nnn { enumext } { package-not-load } { scontents }
25     \RequirePackage{scontents}
26   }
27 }
```

### 13.4 Definition of variables

Variables that do not appear in this section are created by means of `\keys_define:nn` or some function described below.

```

\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int

```

Integer variables will control the nesting levels of the environments and `\anskey` command.

```

28 \int_new:N \l__enumext_level_int
29 \int_new:N \l__enumext_level_h_int
30 \int_new:N \l__enumext_anskey_level_int
31 \int_new:N \l__enumext_keyans_level_int
32 \int_new:N \l__enumext_keyans_level_h_int
33 \int_new:N \l__enumext_keyans_pic_level_int

```

(End of definition for `\l__enumext_level_int` and others.)

```

\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_first_bool
\l__enumext_anskey_env_bool
\l__enumext_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl
\l__enumext_envir_name_tl

```

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_level:` and `\__enumext_keyans_name_and_start:` (§13.5.1).

```

34 \bool_new:N \l__enumext_starred_bool
35 \bool_new:N \g__enumext_starred_bool
36 \bool_new:N \l__enumext_starred_first_bool
37 \bool_new:N \l__enumext_standar_bool
38 \bool_new:N \g__enumext_standar_bool
39 \bool_new:N \l__enumext_standar_first_bool
40 \bool_new:N \l__enumext_anskey_env_bool
41 \bool_new:N \l__enumext_keyans_env_bool
42 \tl_new:N \g__enumext_start_line_tl
43 \tl_new:N \g__enumext_envir_name_tl
44 \tl_new:N \l__enumext_envir_name_tl

```

(End of definition for `\l__enumext_starred_bool` and others.)

```

\l__enumext_counter_i_tl
\l__enumext_counter_ii_tl
\l__enumext_counter_iii_tl
\l__enumext_counter_iv_tl
\l__enumext_counter_v_tl
\l__enumext_counter_vi_tl
\l__enumext_counter_vii_tl
\l__enumext_counter_viii_tl

```

Variables to store the “*name of the counters*” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment. The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments.

The initial values of these variables are set by the function `\__enumext_define_counters:Nn` (§13.11) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§13.14).

```

45 \cs_set_protected:Npn \__enumext_tmp:n #1
46 {
47   \tl_new:c { l__enumext_counter_#1_tl }
48 }
49 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_counter_i_tl` and others.)

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

```

Internal variables used by `ref` key (§13.14).

```

50 \tl_const:Nn \c__enumext_counter_style_tl
51 { { arabic } { roman } { Roman } { alph } { Alph } }
52 \tl_new:N \l__enumext_ref_key_arg_tl
53 \tl_new:N \l__enumext_ref_the_count_tl
54 \cs_set_protected:Npn \__enumext_tmp:n #1
55 {
56   \tl_new:c { l__enumext_renew_the_count_#1_tl }
57   \tl_new:c { l__enumext_the_counter_#1_tl }
58   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
59 }
60 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\c__enumext_counter_style_tl` and others.)

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\l__enumext_resume_name_tl
\l__enumext_resume_active_bool
\g__enumext_starred_series_tl
\g__enumext_standar_series_tl

```

Internal variables used by `resume`, `resume*` and `series` keys (§13.25).

```

61 \int_new:N \g__enumext_resume_int
62 \int_new:N \g__enumext_resume_vii_int
63 \tl_new:N \l__enumext_resume_name_tl
64 \bool_new:N \l__enumext_resume_active_bool
65 \tl_new:N \g__enumext_standar_series_tl
66 \tl_new:N \g__enumext_starred_series_tl

```

(End of definition for `\g__enumext_resume_int` and others.)

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default *label style* and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§13.15) and `label` (§13.13) keys.

```

67 \dim_new:N \l__enumext_current_widest_dim
68 \tl_new:N \g__enumext_counter_styles_tl
69 \tl_new:N \g__enumext_widest_label_tl
70 \box_new:N \l__enumext_label_width_by_box

```

(End of definition for `\l__enumext_current_widest_dim` and others.)

```
\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim
```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§13.18). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace`:NNNNNNNNNN (§13.38.1).

```
71 \cs_set_protected:Npn \__enumext_tmp:n #1
72 {
73   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
74   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
75   \dim_new:c { \l__enumext_leftmargin_#1_dim }
76   \dim_new:c { \l__enumext_itemindent_#1_dim }
77 }
78 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_leftmargin_tmp_X_bool` and others.)

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str
```

Internal variables used by `columns` key (§13.22) and `align` key (§13.13).

```
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \skip_new:c { \l__enumext_multicols_above_#1_skip }
82   \skip_new:c { \l__enumext_multicols_below_#1_skip }
83   \skip_new:c { \g__enumext_multicols_right_#1_skip }
84   \str_new:c { \l__enumext_align_label_pos_#1_str }
85 }
86 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_multicols_above_X_skip` and others.)

```
\g__enumext_minipage_stat_int
\l__enumext_minipage_temp_skip
\l__enumext_minipage_left_skip
\l__enumext_minipage_right_skip
\l__enumext_minipage_after_skip
\g__enumext_minipage_right_skip
\g__enumext_minipage_after_skip
\l__enumext_minipage_left_X_dim
\l__enumext_minipage_active_X_bool
```

Internal variables used by `\miniright` command (§13.23.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.21, §13.23).

```
87 \int_new:N \g__enumext_minipage_stat_int
88 \skip_new:N \l__enumext_minipage_temp_skip
89 \skip_new:N \l__enumext_minipage_left_skip
90 \skip_new:N \l__enumext_minipage_right_skip
91 \skip_new:N \l__enumext_minipage_after_skip
92 \skip_new:N \g__enumext_minipage_right_skip
93 \skip_new:N \g__enumext_minipage_after_skip
94 \cs_set_protected:Npn \__enumext_tmp:n #1
95 {
96   \dim_new:c { \l__enumext_minipage_left_#1_dim }
97   \bool_new:c { \l__enumext_minipage_active_#1_bool }
98 }
99 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\g__enumext_minipage_stat_int` and others.)

```
\l__enumext_wrap_label_X_bool
\l__enumext_wrap_label_opt_X_bool
\l__enumext_start_X_int
\l__enumext_fake_item_indent_X_tl
\l__enumext_label_fill_left_X_tl
\l__enumext_label_fill_right_X_tl
\l__enumext_vspace_a_star_X_bool
\l__enumext_vspace_b_star_X_bool
```

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.18.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§13.13). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.20).

```
100 \cs_set_protected:Npn \__enumext_tmp:n #1
101 {
102   \bool_new:c { \l__enumext_wrap_label_#1_bool }
103   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
104   \int_new:c { \l__enumext_start_#1_int }
105   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
106   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
107   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
108   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
109   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
110 }
111 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_wrap_label_X_bool` and others.)

```

\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_anskey_env_tl
\l__enumext_store_anskey_opt_tl
\l__enumext_store_current_label_tl
\l__enumext_store_current_opt_arg_tl
\l__enumext_store_current_label_tmp_tl

```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§13.26.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_name_tl` saves the  $\{\langle store\ name\rangle\}$  set by the `save-ans` key of the *sequence* and *prop list* in which we will store, the variable `\g__enumext_store_name_tl` it's just a global copy of  $\{\langle store\ name\rangle\}$  used by different functions.

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§13.30) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the  $\langle body\rangle$  and the  $\langle keys\rangle$  of the environment `anskey*` (§13.31).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.37) and `\anspic*` (§13.42.2) for the `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_current_label_tmp_tl` is a temporary variable used by `keyans`, `keyans*` and `keyanspic` at various points.

```

112 \bool_new:N \l__enumext_store_active_bool
113 \tl_new:N \l__enumext_store_name_tl
114 \tl_new:N \g__enumext_store_name_tl
115 \tl_new:N \l__enumext_store_anskey_arg_tl
116 \tl_new:N \l__enumext_store_anskey_env_tl
117 \tl_new:N \l__enumext_store_anskey_opt_tl
118 \tl_new:N \l__enumext_store_current_label_tl
119 \tl_new:N \l__enumext_store_current_opt_arg_tl
120 \tl_new:N \l__enumext_store_current_label_tmp_tl

```

(End of definition for `\l__enumext_store_active_bool` and others.)

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

Internal variables used by the command `\setenumext` (§13.48).

```

121 \tl_new:N \l__enumext_setkey_tmpa_tl
122 \tl_new:N \l__enumext_setkey_tmpb_tl
123 \int_new:N \l__enumext_setkey_tmpa_int
124 \seq_new:N \l__enumext_setkey_tmpa_seq
125 \seq_new:N \l__enumext_setkey_tmpb_seq

```

(End of definition for `\l__enumext_setkey_tmpa_tl` and others.)

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

```

126 \tl_new:N \l__enumext_meta_path_tl
127 \seq_new:N \l__enumext_foreach_print_seq
128 \tl_new:N \l__enumext_foreach_name_prop_tl
129 \tl_new:N \g__enumext_foreach_default_keys_tl

```

(End of definition for `\l__enumext_meta_path_tl` and others.)

```

\l__enumext_print_keyans_starred_tl
\l__enumext_print_keyans_star_bool
\l__enumext_mark_position_str
\g__enumext_item_symbol_aux_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool
\l__enumext_store_upper_level_X_bool

```

Internal variables used by command `\printkeyans` (§13.47), `show-pos` key (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.2) and “*storing structure*”.

```

130 \tl_new:N \l__enumext_print_keyans_starred_tl
131 \bool_new:N \l__enumext_print_keyans_star_bool
132 \str_new:N \l__enumext_mark_position_str
133 \tl_new:N \g__enumext_item_symbol_aux_tl
134 \cs_set_protected:Npn \__enumext_tmp:n #1
135 {
136   \tl_new:c { \l__enumext_print_keyans_#1_tl }
137   \tl_new:c { \l__enumext_store_save_key_#1_tl }
138   \bool_new:c { \l__enumext_store_save_key_#1_bool }
139   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
140 }
141 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_print_keyans_starred_tl` and others.)

```

\l__enumext_anspic_args_seq
\l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
\l__enumext_anspic_label_above_bool
\l__enumext_anspic_mini_pos_str
\g__enumext_keyans_pic_parsep_skip
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim

```

Internal variables used by `keyanspic` environment and `\anspic` command (§13.42.1).

```

142 \seq_new:N \l__enumext_anspic_args_seq
143 \dim_new:N \l__enumext_anspic_mini_width_dim
144 \int_new:N \l__enumext_anspic_above_int
145 \int_new:N \l__enumext_anspic_below_int
146 \bool_new:N \l__enumext_anspic_label_above_bool
147 \str_new:N \l__enumext_anspic_mini_pos_str
148 \skip_new:N \g__enumext_keyans_pic_parsep_skip
149 \box_new:N \l__enumext_anspic_label_box
150 \box_new:N \l__enumext_anspic_body_box
151 \dim_new:N \l__enumext_anspic_label_htdp_dim
152 \dim_new:N \l__enumext_anspic_body_htdp_dim

```

(End of definition for `\l__enumext_anspic_args_seq` and others.)

Internal variables used by “*internal check answer*” mechanism (§13.26.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

153 \bool_new:N \l__enumext_check_answers_bool
154 \bool_new:N \g__enumext_check_ans_key_bool
155 \tl_new:N \l__enumext_check_start_line_env_tl
156 \int_new:N \g__enumext_check_starred_cmd_int
157 \int_new:N \g__enumext_item_anskey_int
158 \int_new:N \g__enumext_item_number_int
159 \bool_new:N \l__enumext_item_number_bool
160 \int_new:N \g__enumext_item_answer_diff_int

```

(End of definition for `\l__enumext_check_answers_bool` and others.)

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§13.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

161 \bool_new:N \l__enumext_hyperref_bool
162 \bool_new:N \l__enumext_footnotes_key_bool

```

(End of definition for `\l__enumext_hyperref_bool` and `\l__enumext_footnotes_key_bool`.)

Internal variables used by `save-ref` key (§13.27). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the `⟨labels⟩` defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `\__enumext_newlabel:nn` (§13.7) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

163 \tl_new:N \l__enumext_newlabel_arg_one_tl
164 \tl_new:N \l__enumext_newlabel_arg_two_tl
165 \tl_new:N \l__enumext_write_aux_file_tl
166 \cs_set_protected:Npn \__enumext_tmp:n #1
167 {
168   \tl_new:c { l__enumext_label_copy_#1_tl }
169 }
170 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_newlabel_arg_one_tl` and others.)

Internal variables used for redefinition of `\footnote` (§13.8).

```

171 \int_new:N \g__enumext_footnote_standar_int
172 \int_new:N \g__enumext_footnote_starred_int
173 \seq_new:N \g__enumext_footnote_standar_arg_seq
174 \seq_new:N \g__enumext_footnote_starred_arg_seq
175 \seq_new:N \g__enumext_footnote_standar_int_seq
176 \seq_new:N \g__enumext_footnote_starred_int_seq

```

(End of definition for `\g__enumext_footnote_standar_int` and others.)

Internal variables used by `enumext*` and `keyans*` environments.

```

177 \cs_set_protected:Npn \__enumext_tmp:n #1
178 {
179   \bool_new:c { l__enumext_item_starred_#1_bool }
180   \int_new:c { l__enumext_item_column_pos_#1_int }
181   \int_new:c { g__enumext_item_count_all_#1_int }
182   \int_new:c { l__enumext_joined_item_#1_int }
183   \int_new:c { l__enumext_joined_item_aux_#1_int }
184   \int_new:c { l__enumext_tmpa_#1_int }
185   \dim_new:c { l__enumext_tmpa_#1_dim }
186   \box_new:c { l__enumext_item_text_#1_box }
187   \dim_new:c { l__enumext_joined_width_#1_dim }
188   \dim_new:c { l__enumext_item_width_#1_dim }
189   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
190   \str_new:c { l__enumext_align_label_#1_str }
191   \bool_new:c { g__enumext_minipage_active_#1_bool }
192   \box_new:c { l__enumext_miniright_code_#1_box }
193   \bool_new:c { g__enumext_minipage_center_#1_bool }
194   \dim_new:c { g__enumext_minipage_right_#1_dim }
195   \skip_new:c { g__enumext_minipage_right_#1_skip }
196 }
197 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_item_starred_X_bool` and others.)

`\c__enumext_all_envs_clist` An internal `clist-var` variable to run with `\__enumext_tmp:n`.

```
198 \clist_const:Nn \c__enumext_all_envs_clist
199 {
200   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
201   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
202 }
```

(End of definition for `\c__enumext_all_envs_clist`.)

### 13.5 Some utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

`\seq_use:NV`

```
203 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
204 \cs_generate_variant:Nn \seq_use:Nn { NV }
```

(End of definition for `\keys_precompile:neN` and `\seq_use:NV`.)

`\__enumext_at_begin_document:n` A internal “hook” function used for copying plain `list` and `minipage` environments definition and `hyperref` detection.

```
205 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
206 {
207   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
208 }
```

(End of definition for `\__enumext_at_begin_document:n`.)

`\__enumext_after_env:nn` A internal “hook” functions for execute code `mini-right` and `mini-right*` keys outside the `enumext*` and `keyans*` environments and print `check-ans` outside the `enumext` and `enumext*` environments.

`\__enumext_before_env:nn`

```
209 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
210 {
211   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
212 }
213 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
214 {
215   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
216 }
```

(End of definition for `\__enumext_after_env:nn` and `\__enumext_before_env:nn`.)

`\__enumext_level:` Function for check current level in `enumext`.

```
217 \cs_new:Nn \__enumext_level:
218 {
219   \int_to_roman:n { \l__enumext_level_int }
220 }
```

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

`\__enumext_if_is_int:nT` A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys.

`\__enumext_if_is_int:nF`

`\__enumext_if_is_int:nTF`

This function is taken directly from the answer given by Henri Menke in [How to test if an expl3 function argument is an integer expression?](#).

```
221 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
222 {
223   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
224   { \prg_return_true: }
225   { \prg_return_false: }
226 }
```

(End of definition for `\__enumext_if_is_int:nT`, `\__enumext_if_is_int:nF`, and `\__enumext_if_is_int:nTF`.)

`\__enumext_regex_counter_style:`

The internal function `\__enumext_regex_counter_style:` replace the ‘`*`’ with the actual counter of the running level and is used by the `ref` key. It loops through the defined counter styles in `\c__enumext_counter_style_tl` and replace ‘`*`’ by real command, for example, looking for `\arabic*` and replacing that by `\arabic{<counter>}` defined on the current level.

```
227 \cs_new_protected:Nn \__enumext_regex_counter_style:
228 {
229   \tl_map_inline:Nn \c__enumext_counter_style_tl
230   {
231     \regex_replace_once:nnN { \c{##1}\* }
232     { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
233   }
234 }
```



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

`\__enumext_show_length:nnn`

Internal function used by `show-length` key to show “*all lengths*” calculated and use in `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

235 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
236 {
237     * ~ #2
238     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
239     = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
240 }

```

(End of definition for `\__enumext_show_length:nnn`.)

`\__enumext_unskip_unkern:`

The function `\__enumext_unskip_unkern:` will remove the last *⟨skip⟩* or *⟨kern⟩* at execution time using the values `11` and `12` of `\lastnodetype` to apply `\unskip` or `\unkern` according to the case.

```

241 \cs_new_protected:Nn \__enumext_unskip_unkern:
242 {
243     \int_case:nnT { \lastnodetype }
244     {
245         { 11 } { \unskip }
246         { 12 } { \unkern }
247     }
248 }

```

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

### 13.5.1 Utilities for environments and levels

`\__enumext_is_not_nested:`

The function `\__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “*true*” only if the environments `enumext` and `enumext*` are NOT nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

`\__enumext_is_on_first_level:`

```

249 \cs_new_protected:Nn \__enumext_is_not_nested:
250 {
251     \str_case:en { \@currenvir }
252     {
253         {enumext}
254         {
255             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
256             \bool_lazy_and:nnT
257             { \bool_not_p:n { \g__enumext_standar_bool } }
258             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
259             {
260                 \bool_gset_true:N \g__enumext_standar_bool
261             }
262         }
263         {enumext*}
264         {
265             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
266             \bool_lazy_and:nnT
267             { \bool_not_p:n { \g__enumext_starred_bool } }
268             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
269             {
270                 \bool_gset_true:N \g__enumext_starred_bool
271             }
272         }
273     }
274 }

```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.26.1), `\l__enumext_starred_first_bool` (§13.26.1) and `\l__enumext_anskey_env_bool` (§13.31) to “*true*” only if the environment is not nested and we are in the “*first level*” of it . We will also save the *start line number* of each environment in the variable `\g__enumext_start_line_tl` and the *name* of each environment in the variable `\g__enumext_envir_name_tl` to use in messages related to the `check-ans` key and `.log` file.

```

275 \cs_new_protected:Nn \__enumext_is_on_first_level:
276 {
277     \bool_lazy_all:nT
278     {
279         { \bool_if_p:N \g__enumext_standar_bool }
280         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
281         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
282     }

```

```

283     {
284       \bool_set_true:N \l__enumext_standar_first_bool
285       \bool_set_true:N \l__enumext_anskey_env_bool
286       \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
287       \tl_gset:Ne \g__enumext_start_line_tl
288         {
289           on ~ line ~ \exp_not:V \inputlineno
290         }
291     }
292 \bool_lazy_all:nT
293 {
294   { \bool_if_p:N \g__enumext_starred_bool }
295   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
296   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
297 }
298 {
299   \bool_set_true:N \l__enumext_starred_first_bool
300   \bool_set_true:N \l__enumext_anskey_env_bool
301   \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
302   \tl_gset:Ne \g__enumext_start_line_tl
303     {
304       on ~ line ~ \exp_not:V \inputlineno
305     }
306 }
307 }

```

(End of definition for `\__enumext_is_not_nested:` and `\__enumext_is_on_first_level:`)

`\__enumext_keyans_name_and_start:`

The function `\__enumext_keyans_name_and_start:` will save the start line number and name of the environments `keyans`, `keyans*` and `keyanspic` in the variables `\l__enumext_check_start_line_env_tl` and `\l__enumext_envir_name_tl` to use in the `\__enumext_check_starred_cmd:n` function.

```

308 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
309 {
310   \str_case:en { \@currenvir }
311   {
312     {keyans}
313     {
314       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
315       \tl_set:Ne \l__enumext_check_start_line_env_tl
316         {
317           in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
318         }
319     }
320     {keyans*}
321     {
322       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
323       \tl_set:Ne \l__enumext_check_start_line_env_tl
324         {
325           in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
326         }
327     }
328     {keyanspic}
329     {
330       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
331       \tl_set:Ne \l__enumext_check_start_line_env_tl
332         {
333           in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
334         }
335     }
336   }
337 }

```

(End of definition for `\__enumext_keyans_name_and_start:`)

### 13.5.2 Utilities for log and terminal

The function `\__enumext_reset_global_vars:` will be passed to the function `\__enumext_execute_after_env:` and will return the global variables to their default values after being used.

```

338 \cs_new_protected:Nn \__enumext_reset_global_vars:
339 {
340   \__enumext_reset_global_int:
341   \__enumext_reset_global_bool:

```

```

342   \__enumext_reset_global_tl:
343 }
344 \cs_new_protected:Nn \__enumext_reset_global_int:
345 {
346   \int_gzero:N \g__enumext_item_number_int
347   \int_gzero:N \g__enumext_item_anskey_int
348   \int_gzero:N \g__enumext_item_answer_diff_int
349 }
350 \cs_new_protected:Nn \__enumext_reset_global_bool:
351 {
352   \bool_gset_false:N \g__enumext_check_ans_key_bool
353   \bool_gset_false:N \g__enumext_standar_bool
354   \bool_gset_false:N \g__enumext_starred_bool
355 }
356 \cs_new_protected:Nn \__enumext_reset_global_tl:
357 {
358   \tl_gclear:N \g__enumext_store_name_tl
359   \tl_gclear:N \g__enumext_start_line_tl
360   \tl_gclear:N \g__enumext_envir_name_tl
361 }

```

(End of definition for `\__enumext_reset_global_vars:` and others.)

`\__enumext_log_global_vars:` The function `\__enumext_log_global_vars:` will be passed to the function `\__enumext_execute_after_env:` and write to the `.log` file the number of elements saved in the *prop list* and *sequence* created by the *save-ans* key along with the value of the integer variable created for the *resume* key.

```

362 \cs_new_protected:Nn \__enumext_log_global_vars:
363 {
364   \msg_log:nneeee { enumext } { prop-seq-int-hook }
365   { \g__enumext_store_name_tl }
366   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
367   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
368   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
369 }

```

The function `\__enumext_log_answer_vars:` will be passed to the function `\__enumext_execute_after_env:` and write to the `.log` file the number of items and answers along with the difference between them.

```

370 \cs_new_protected:Nn \__enumext_log_answer_vars:
371 {
372   \msg_log:nneeee { enumext } { item-answer-hook }
373   { \int_use:N \g__enumext_item_number_int }
374   { \int_use:N \g__enumext_item_anskey_int }
375   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
376 }

```

(End of definition for `\__enumext_log_global_vars:` and `\__enumext_log_answer_vars:`.)

## 13.6 Copying list and minipage environments

The `list` environment provided by  $\text{\LaTeX}$  has the following plain form:

```

\list{⟨arg one⟩}{⟨arg two⟩}
  \item[⟨opt⟩]
\endlist

```

And `minipage` environment provided by  $\text{\LaTeX}$  has the following (simplified) plain form:

```

\minipage[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}
  ⟨internal implement⟩
\endminipage

```

As a precaution we copy them using `\__enumext_at_begin_document:n` in case any package redefines the `list` environment or a related command.

- For compatibility with *tagged* PDF we should use `\NewCommandCopy` and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `\ltxcmd` (see `latex-lab-block`).

`\__enumext_start_list:nn` The functions `\__enumext_start_list:nn` and `\__enumext_stop_list:` correspond to copies of `\list` and `\endlist` from plain definition of `list` environment, the function `\__enumext_item_std:w` is a copy of the `\item` command.

```

\__enumext_stop_list:
\__enumext_item_std:w
\__enumext_minipage:w
\__enumext_endminipage:
377 \__enumext_at_begin_document:n
378 {
379   \cs_new_eq:NN \__enumext_start_list:nn \list

```

```

380 \cs_new_eq:NN \__enumext_stop_list: \endlist
381 \NewCommandCopy \__enumext_item_std:w \item
382 }

```

The functions `\__enumext_minipage:w` and `\__enumext_endminipage:` correspond to copies of `\minipage` and `\endminipage` from plain definition of `minipage` environment.

```

383 \__enumext_at_begin_document:n
384 {
385 \cs_new_eq:NN \__enumext_minipage:w \minipage
386 \cs_new_eq:NN \__enumext_endminipage: \endminipage
387 }

```

(End of definition for `\__enumext_start_list:nn` and others.)

### 13.7 Compatibility with hyperref and footnotehyper

First we define the necessary rules using “hooks” to determine if the `hyperref` package is loaded.

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:
388 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
389 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

The function `\__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyperref_bool` to “true” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `\l__enumext_footnotes_key_bool` to “true”.

```

390 \cs_new_protected:Nn \__enumext_after_hyperref:
391 {
392 \IfPackageLoadedTF { hyperref }
393 {
394 \msg_info:nnn { enumext } { package-load } { hyperref }
395 \bool_set_true:N \l__enumext_hyperref_bool
396 \IfHyperBoolean{hyperfootnotes}
397 {
398 \bool_set_true:N \l__enumext_footnotes_key_bool
399 }
400 { }
401 }
402 { }

```

If the state of the variable `\l__enumext_footnotes_key_bool` is true we will check if the package `footnotehyper` is loaded, in case it is not present, we will set the value of `\l__enumext_footnotes_key_bool` to false and we will redefine `\footnote`.

```

403 \bool_if:NT \l__enumext_footnotes_key_bool
404 {
405 \IfPackageLoadedTF { footnotehyper }
406 {
407 \msg_info:nnn { enumext } { package-load } { footnotehyper }
408 }
409 {
410 \bool_set_false:N \l__enumext_footnotes_key_bool
411 }
412 }

```

The functions `\__enumext_hypertarget:nn` and `\__enumext_phantomsection:` correspond to the internal copies of `\hypertarget` and `\phantomsection`. If the boolean variable `\l__enumext_hyperref_bool` is false the functions `\__enumext_hypertarget:nn` and `\__enumext_phantomsection:` will be disabled.

```

413 \bool_if:NTF \l__enumext_hyperref_bool
414 {
415 \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
416 \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
417 }
418 {
419 \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
420 \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
421 }
422 }

```

(End of definition for `\__enumext_after_hyperref:`, `\__enumext_hypertarget:nn`, and `\__enumext_phantomsection:`.)

```

\__enumext_newlabel:nn

```

The function `\__enumext_newlabel:nn` write the information to the `.aux` file when using the `save-ref` key. The arguments taken by the function are:

```

#1: \l__enumext_newlabel_arg_one_tl
#2: \l__enumext_newlabel_arg_two_tl

```

- The trick here is to manage the number of arguments passed to `\newlabel{#1}{#2}` according to the presence of the `hyperref` package.

```

423 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
424 {
425   \protected@write \@auxout { }
426   {
427     \token_to_str:N \newlabel {#1}
428     {
429       {#2}
430       \bool_if:NT \l__enumext_hyperref_bool
431       { { \thepage } {#2} {#1} }
432       { }
433     }
434   }
435   \__enumext_hypertarget:nn {#1} { }
436   \__enumext_phantomsection:
437 }

```

(End of definition for `\__enumext_newlabel:nn`.)

### 13.8 Internal redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

To keep the correct numbering of `\footnote` and to make it work correctly in the `enumext*` and `keyans*` environments, and `mini-env` key it is necessary to redefine the `\footnote` command. This implementation is adapted from the answer given by Clea F. Rees (@cfr) in [footnotes in boxes compatible with hyperref](#).

```

438 \cs_new_protected:Npn \__enumext_footnotetext:nn
439 {
440   \footnotetext[#1]{#2}
441 }
442 \cs_new_protected:Npn \__enumext_renew_footnote:
443 {
444   \RenewDocumentCommand \footnote { o +m }
445   {
446     \tl_if_novalue:nTF {##1}
447     {
448       \stepcounter{footnote}
449       \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
450     }
451     {
452       \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
453     }
454     \footnotemark [ \g__enumext_footnote_standar_int ]
455     \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
456     \seq_gput_right:NV \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
457   }
458 }
459 \cs_new_protected:Npn \__enumext_print_footnote:
460 {
461   \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
462   {
463     \seq_map_pairwise_function:NNN
464     \g__enumext_footnote_standar_int_seq
465     \g__enumext_footnote_standar_arg_seq
466     \__enumext_footnotetext:nn
467   }
468   \seq_gclear:N \g__enumext_footnote_standar_arg_seq
469   \seq_gclear:N \g__enumext_footnote_standar_int_seq
470 }
471 \cs_new_protected:Npn \__enumext_renew_footnote_mini:
472 {
473   \RenewDocumentCommand \footnote { o +m }
474   {
475     \tl_if_novalue:nTF {##1}
476     {
477       \stepcounter{footnote}
478       \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
479     }
480     {
481       \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
482     }
483     \footnotemark [ \g__enumext_footnote_starred_int ]

```

```

484         \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
485         \seq_gput_right:NV \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
486     }
487 }
488 \cs_new_protected:Nn \__enumext_print_footnote_mini:
489 {
490     \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
491     {
492         \seq_map_pairwise_function:NNN
493         \g__enumext_footnote_starred_int_seq
494         \g__enumext_footnote_starred_arg_seq
495         \__enumext_footnotetext:nn
496     }
497     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
498     \seq_gclear:N \g__enumext_footnote_starred_int_seq
499 }

```

(End of definition for `\__enumext_footnotetext:nn`, `\__enumext_renew_footnote:`, and `\__enumext_print_footnote:.`)

```

\__enumext_renew_footnote_standar:
\__enumext_print_footnote_standar:
\__enumext_renew_footnote_starred:
\__enumext_print_footnote_starred:

```

We encapsulate the redefinition of `\footnote` to pass it to internal `__enumext_mini_page` environment used by the `mini-env` key in the `enumext` and `keyans` environments. We will run the redefinition when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```

500 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
501 {
502     \bool_if:NT \g__enumext_standar_bool
503     {
504         \IfDocumentMetadataTF
505         {
506             \__enumext_renew_footnote:
507         }
508         {
509             \bool_if:NF \l__enumext_footnotes_key_bool
510             {
511                 \__enumext_renew_footnote:
512             }
513         }
514     }
515 }
516 \cs_new_protected:Nn \__enumext_print_footnote_standar:
517 {
518     \bool_if:NT \g__enumext_standar_bool
519     {
520         \IfDocumentMetadataTF
521         {
522             \__enumext_print_footnote:
523         }
524         {
525             \bool_if:NF \l__enumext_footnotes_key_bool
526             {
527                 \__enumext_print_footnote:
528             }
529         }
530     }
531 }

```

We encapsulate the redefinition of `\footnote` to pass it to the `enumext*` and `keyans*` environments. We will run the redefinition when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```

532 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
533 {
534     \IfDocumentMetadataTF
535     {
536         \__enumext_renew_footnote_mini:
537     }
538     {
539         \bool_if:NF \l__enumext_footnotes_key_bool
540         {
541             \__enumext_renew_footnote_mini:
542         }
543     }
544 }
545 \cs_new_protected:Nn \__enumext_print_footnote_starred:

```



```

546 {
547   \IfDocumentMetadataTF
548   {
549     \__enumext_print_footnote_mini:
550   }
551   {
552     \bool_if:NF \l__enumext_footnotes_key_bool
553     {
554       \__enumext_print_footnote_mini:
555     }
556   }
557 }
558 \__enumext_after_env:nn { enumext* }
559 {
560   \__enumext_print_footnote_starred:
561 }
562 \__enumext_after_env:nn { keyans* }
563 {
564   \__enumext_print_footnote_starred:
565 }

```

(End of definition for \\_\_enumext\_renew\_footnote\_standar: and others.)

### 13.9 The internal minipage environment

```

\__enumext_internal_mini_page:
  __enumext_mini_env*

```

The function `\__enumext_internal_mini_page:` creates a internal `\__enumext_mini_page` environment (custom version of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\skip_vertical:N \c_zero_skip` to maintain alignment on “top” in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces “below”. This environment will be used internally by the `mini-env` key, it is NOT documented in the user interface and is for internal use only. Within this environment we redefine `\footnote` to make them look the same as if they were elsewhere in the document. This function is passed to the function `\__enumext_safe_exec:` in the `enumext` environment definition (§13.39) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.44)

```

566 \cs_new_protected:Nn \__enumext_internal_mini_page:
567 {
568   \int_compare:nNt { \l__enumext_level_int } = { 0 }
569   {
570     \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
571     {
572       \__enumext_renew_footnote_standar:
573       \__enumext_minipage:w [ t ] { ##1 }
574       \legacy_if_gset_false:n { @minipage }
575       \skip_vertical:N \c_zero_skip
576     }
577     {
578       \skip_vertical:N \c_zero_skip
579       \__enumext_endminipage:
580       \__enumext_print_footnote_standar:
581     }
582   }
583 }

```

(End of definition for \\_\_enumext\_internal\_mini\_page: and \_\_enumext\_mini\_env\*.)

### 13.10 Definition of public dimension

The package `enumext` only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. This dimension is set in all environments and is only used by the `wrap-ans` key at its default value.

```

584 \dim_zero_new:N \itemwidth

```

### 13.11 Definition of counters

```

\__enumext_define_counters:Nn
  enumXi
  enumXii
  enumXiii
  enumXiv
  enumXv
  enumXvi
  enumXvii
  enumXviii

```

To create the necessary “counters” we must first make sure that they are not already defined by the user or a package such as `enumitem`, otherwise a error will be returned and the package loading will be aborted. The arguments taken by the function are:

- #1: A token list `\l__enumext_counter_X_tl` for “store” the counter’s name.
- #2: The counter’s name.

```

585 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
586 {
587   \cs_if_exist:cTF { c@ #2 }
588   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
589   {
590     \tl_set:Nn #1 { #2 }
591     \newcounter { #2 }
592   }
593 }

```

The counters created here are `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment, `enumXvi` for `keyanspic` environment, `enumXvii` for `enumext*` and `enumXviii` for the `keyans*` environments.

```

594 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
595 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
596 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
597 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
598 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
599 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
600 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
601 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

(End of definition for `\__enumext_define_counters:Nn` and others.)

### 13.12 Definition of labels

This part of the code is inspired by the `enumitem` package. The idea is to be able to access the counters using `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

```
\__enumext_register_counter_style:Nn
```

These `<counters>` will be used as default `<labels>` if the `label` key is not used for the different levels of the `enumext`, `enumext*`, `keyans` and `keyans*` environments, so it is necessary to get a default value for `labelwidth` from these `<labels>` at the same time.

```

602 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
603 {
604   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
605   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
606 }
607 \__enumext_register_counter_style:Nn \arabic { 0 }
608 \__enumext_register_counter_style:Nn \Alph { M }
609 \__enumext_register_counter_style:Nn \alph { m }
610 \__enumext_register_counter_style:Nn \Roman { VIII }
611 \__enumext_register_counter_style:Nn \roman { viii }

```

(End of definition for `\__enumext_register_counter_style:Nn`.)

```
\__enumext_label_width_by_box:Nn
```

```
\__enumext_label_width_by_box:cv
```

The function `\__enumext_label_width_by_box:Nn` set the default `\labelwidth` using a box width if no `labelwidth` key is passed.

```

612 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
613 {
614   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
615   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
616 }
617 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }

```

(End of definition for `\__enumext_label_width_by_box:Nn`.)

```
\__enumext_label_style:Nnn
```

```
\__enumext_label_style:cvn
```

The function `\__enumext_label_style:Nnn` is used by the `label` key to creates the variables containing the `<label style>` and will allow to use `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` as arguments. It loops through the defined counter styles in `\g__enumext_counter_styles_tl` (`\arabic`, `\alph`, `\Alph`, `\roman`, and `\Roman`) for example, looking for `\roman*` and replacing that by `\roman{<counter>}`, and doing the same for the `\g__enumext_widest_label_tl` to keep both in sync.

```

618 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
619 {
620   \tl_clear_new:N #1
621   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
622   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
623   \tl_map_inline:Nn \g__enumext_counter_styles_tl
624   {
625     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
626     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
627     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }

```

```

628     }
629     \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
630     { \tl_use:N \g__enumext_widest_label_tl }
631     \tl_set_eq:cN { the #2 } #1
632   }
633   \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

(End of definition for \\_\_enumext\_label\_style:Nnn.)

### 13.13 Setting keys associated with label

When *tagged* PDF is active `\makelabel` is redefined using `\makebox` to work correctly (§13.34). From the user side it is convenient to have a key that allows using this redefinition with `\makebox` without having `\IfDocumentMetadataTF` active.

`mode-box` We define the key `mode-box` only for the “first level” of `enumext` and `enumext*` environments.

```

634 \cs_set_protected:Npn \__enumext_tmp:n #1
635 {
636   \keys_define:nn { enumext / #1 }
637   {
638     mode-box .bool_set:N = \l__enumext_mode_box_bool,
639     mode-box .initial:n = false,
640     mode-box .value_forbidden:n = true,
641   }
642 }
643 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for `mode-box`.)

`font` `labelsep` `labelwidth` `wrap-label` `wrap-label*` Definition of keys `font`, `labelsep`, `labelwidth`, `wrap-label` and `wrap-label*` keys for `enumext` and `keyans` environments.

```

644 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
645 {
646   \keys_define:nn { enumext / #1 }
647   {
648     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
649     font .value_required:n = true,
650     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
651     labelsep .initial:n = {0.3333em},
652     labelsep .value_required:n = true,
653     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
654     labelwidth .value_required:n = true,
655     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
656     wrap-label .initial:n = {##1},
657     wrap-label .value_required:n = true,
658     wrap-label* .code:n = {
659       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
660       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
661     },
662     wrap-label* .value_required:n = true,
663   }
664 }
665 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `font` and others.)

`align` The `align` key is implemented differently for “starred” and “non starred” environments. For compatibility with *tagged* PDF we must set `\l__enumext_align_label_pos_X_str`.

```

666 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
667 {
668   \keys_define:nn { enumext / #1 }
669   {
670     align .choice:,
671     align / left .code:n =
672       {
673         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
674         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
675         \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
676       },
677     align / right .code:n =
678       {
679         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }

```

```

680         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
681         \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
682     },
683     align / center .code:n =
684     {
685         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
686         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
687         \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
688     },
689     align / unknown .code:n =
690         \msg_error:nneee { enumext } { unknown-choice }
691         { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
692     align .initial:n = left,
693     align .value_required:n = true,
694 }
695 }
696 \clist_map_inline:nn
697 {
698     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
699 }
700 { \__enumext_tmp:nn #1 }

701 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
702 {
703     \keys_define:nn { enumext / #1 }
704     {
705         align .choice:,
706         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
707         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
708         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
709         align / unknown .code:n =
710             \msg_error:nneee { enumext } { unknown-choice }
711             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
712         align .initial:n = left,
713         align .value_required:n = true,
714     }
715 }
716 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for *align*.)

### 13.14 Setting label and ref keys

The implementation of the keys `label` and `ref` are part of the core of the package `enumext`, here the default values for  $\langle label \rangle$ , the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “*label and ref*” system.

#### 13.14.1 Define and set label and ref keys for enumext environment

Here we set the default  $\langle labels \rangle$  of the *four levels* of `enumext` environment, along with the default value for `labelwidth` key and `ref` key.

```

label
ref
\__enumext_label_i_tl
\__enumext_label_ii_tl
\__enumext_label_iii_tl
\__enumext_label_iv_tl

717 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
718 {
719     \keys_define:nn { enumext / #1 }
720     {
721         label .code:n = {
722             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
723             { l__enumext_counter_#2_tl } {##1}
724             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
725             \l__enumext_current_widest_dim
726         },
727         label .initial:n = #3,
728         label .value_required:n = true,
729         ref .code:n = \__enumext_standar_ref:n {##1},
730         ref .value_required:n = true,
731     }
732 }
733 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
734 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
735 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
736 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

(End of definition for *label* and *others*.)

\\_\_enumext\_standar\_ref:n  
 \\_\_enumext\_standar\_ref:

The \\_\_enumext\_standar\_ref:n first we will pass the key argument to \l\_\_enumext\_ref\_key\_arg\_tl and we will analyze its state, if it is not *empty* we will make a copy of the current counter in \l\_\_enumext\_ref\_the\_count\_tl and we will execute the function \\_\_enumext\_regex\_counter\_style: which will return the modified \l\_\_enumext\_ref\_key\_arg\_tl and we make the value of \l\_\_enumext\_ref\_the\_count\_tl the same as that \l\_\_enumext\_the\_counter\_X\_tl which contains \theenumX and finally we set \l\_\_enumext\_renew\_the\_count\_X\_tl with the renewed command.

```

737 \cs_new_protected:Npn \__enumext_standar_ref:n #1
738 {
739   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
740   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
741   {
742     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
743   }
744   {
745     \tl_set_eq:Nc
746     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
747     \__enumext_regex_counter_style:
748     \tl_set_eq:Nc
749     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
750     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
751     {
752       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
753     }
754   }
755 }

```

Finally the function \\_\_enumext\_standar\_ref: will execute the modification for the reference system in the second argument of the environment definition `enumext`.

```

756 \cs_new_protected:Nn \__enumext_standar_ref:
757 {
758   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
759   {
760     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
761   }
762 }

```

(End of definition for \\_\_enumext\_standar\_ref:n and \\_\_enumext\_standar\_ref:.)

### 13.14.2 Define and set label and ref keys for enumext\* and keyans\* environments

Here we set the default *labels* for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key and `ref` key.

```

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
763 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
764 {
765   \keys_define:nn { enumext / #1 }
766   {
767     label .code:n = {
768       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
769       { \l__enumext_counter_#2_tl } {##1}
770       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
771       \l__enumext_current_widest_dim
772     },
773     label .initial:n = #3,
774     label .value_required:n = true,
775     ref .code:n = \__enumext_starred_ref:n {##1},
776     ref .value_required:n = true,
777   }
778 }
779 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
780 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

(End of definition for label and others.)

\\_\_enumext\_starred\_ref:n  
 \\_\_enumext\_starred\_ref:

The implementation of \\_\_enumext\_starred\_ref:n is the same as that used for the environment `enumext`.

```

781 \cs_new_protected:Npn \__enumext_starred_ref:n #1
782 {
783   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
784   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
785   {
786     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
787     {
788       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }

```

```

789     }
790     {
791         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
792         \__enumext_regex_counter_style:
793         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
794         \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
795         {
796             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
797         }
798     }
799 }
800 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
801 {
802     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
803     {
804         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
805     }
806     {
807         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
808         \__enumext_regex_counter_style:
809         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
810         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
811         {
812             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
813         }
814     }
815 }
816 }

```

Finally the function `\__enumext_starred_ref:` will execute the modification for the reference system in the second argument of the `enumext*` and `keyans*` environment definition.

```

817 \cs_new_protected:Nn \__enumext_starred_ref:
818 {
819     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
820     {
821         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
822         {
823             \tl_use:N \l__enumext_renew_the_count_vii_tl
824         }
825     }
826     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
827     {
828         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
829         {
830             \tl_use:N \l__enumext_renew_the_count_viii_tl
831         }
832     }
833 }

```

(End of definition for `\__enumext_starred_ref:n` and `\__enumext_starred_ref:`)

### 13.14.3 Define and set label and ref keys for keyans and keyanspic environments

Here we set the default `<label>` for `keyans` and `keyanspic` environment, along with the default value for `labelwidth` and `ref` key. The `keyanspic` environment use the same `<label>` as the `keyans` environment.

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
834 \keys_define:nn { enumext / keyans }
835 {
836     label .code:n = {
837         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
838         { \l__enumext_counter_v_tl } {#1}
839         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
840         \l__enumext_current_widest_dim
841         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
842         { \l__enumext_counter_vi_tl } {#1}
843         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
844         \l__enumext_current_widest_dim
845     },
846     label .initial:n = \Alph*,
847     label .value_required:n = true,
848     ref .code:n = \__enumext_keyans_ref:n {#1},
849     ref .value_required:n = true,
850 }

```



(End of definition for `label` and others.)

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:
851 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
852 {
853   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
854   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
855   {
856     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
857   }
858   {
859     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
860     \__enumext_regex_counter_style:
861     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
862     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
863     {
864       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
865     }
866   }
867 }

```

Finally the function `\__enumext_keyans_ref:` will execute the modification for the reference system in the second argument of the `keyans*` environment definition.

```

868 \cs_new_protected:Nn \__enumext_keyans_ref:
869 {
870   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
871   {
872     \tl_use:N \l__enumext_renew_the_count_v_tl
873   }
874 }

```

(End of definition for `\__enumext_keyans_ref:n` and `\__enumext_keyans_ref:`.)

### 13.15 Setting start, start\* and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn
\__enumext_start_from:cce

```

The function `\__enumext_start_from:NNn` used by `start` and `start*` keys take three arguments:

#1: `\l__enumext_label_X_tl`  
 #2: `\l__enumext_start_X_int`  
 #3: *<integer or string>*

The first argument of this function are the “*counter style*” set by `label` key, the second argument is returned by the function, the third argument can be an *<integer>* or *<string>* of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start=A` or `start=1` to be used.

```

875 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
876 {
877   \__enumext_if_is_int:nTF { #3 }
878   {
879     \int_set:Nn #2 {#3}
880   }
881   {
882     \regex_match:nVT { \c{Alph} | \c{alph} } { #1 }
883     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
884     \regex_match:nVT { \c{Roman} | \c{roman} } { #1 }
885     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
886   }
887 }
888 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

(End of definition for `\__enumext_start_from:NNn`.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

The function `\__enumext_widest_from:nNNn` used by the `widest` key take four arguments:

#1: The counter associated with the environment level  
 #2: `\l__enumext_label_X_tl`  
 #3: `\l__enumext_labelwidth_X_dim`  
 #4: *<integer or string>*

The second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an *<integer>* or *<string>* of the form `\Alph`, `\alph`, `\Roman` or `\roman`. The value of the four argument is set temporarily for the identified counter in this point (level), then the value is expanded into a “*box*” and the “*width*” of the “*box*” is returned.

```

889 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
890 {
891   \__enumext_if_is_int:nTF {#4}

```

```

892     {
893         \setcounter{enumX#1} { #4 }
894     }
895     {
896         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
897         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
898         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
899         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
900     }
901     \__enumext_label_width_by_box:cv
902     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
903 }
904 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for \\_\_enumext\_widest\_from:nNNn.)

start\* Now define and set start\*, start and widest keys for enumext, enumext\*, keyans and keyans\* environments.

```

905 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
906 {
907     \keys_define:nn { enumext / #1 }
908     {
909         start* .code:n = {
910             \__enumext_start_from:ccn
911             { l__enumext_label_#2_tl }
912             { l__enumext_start_#2_int } {##1}
913         },
914         start* .value_required:n = true,
915         start .code:n = {
916             \__enumext_start_from:cce
917             { l__enumext_label_#2_tl }
918             { l__enumext_start_#2_int } { \int_eval:n {##1} }
919         },
920         start .initial:n = 1,
921         start .value_required:n = true,
922         widest .code:n = {
923             \__enumext_widest_from:nccn {#2}
924             { l__enumext_label_#2_tl }
925             { l__enumext_labelwidth_#2_dim } {##1}
926         },
927         widest .value_required:n = true,
928     }
929 }
930 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for start, start\*, and widest.)

### 13.16 Setting keys for vertical spaces

topsep Define and set topsep, partopsep, parsep, itemsep, noitemsep and nosep keys for enumext, enumext\*, keyans and keyans\* environments.

```

931 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
932 {
933     \keys_define:nn { enumext / #1 }
934     {
935         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
936         topsep .initial:n = {#3},
937         topsep .value_required:n = true,
938         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
939         partopsep .initial:n = {#4},
940         partopsep .value_required:n = true,
941         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
942         parsep .initial:n = {#5},
943         parsep .value_required:n = true,
944         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
945         itemsep .initial:n = {#6},
946         itemsep .value_required:n = true,
947         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
948         noitemsep .value_forbidden:n = true,
949         nosep .meta:n = {
950             itemsep = 0pt, parsep = 0pt,

```

```

951             topsep = 0pt, partopsep = 0pt,
952             },
953     nosep      .value_forbidden:n = true,
954   }
955 }

```

Now we set the values based on standard `article` class in 10pt.

```

956 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
957   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
958   { 4.0pt plus 2.0pt minus 1.0pt }
959 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
960   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
961   { 2.0pt plus 1.0pt minus 1.0pt }
962 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
963   { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
964 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
965   { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
966 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
967   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
968   { 2.0pt plus 1.0pt minus 1.0pt }
969 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
970   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
971   { 4.0pt plus 2.0pt minus 1.0pt }
972 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
973   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
974   { 2.0pt plus 1.0pt minus 1.0pt }

```

(End of definition for `topsep` and others.)

### 13.17 Setting base-fix key

When nesting starting right after `\item` (without material between them) there is a problem with the alignment of the *baseline* between the two environments. One way to get around this problem is to place `\mode_leave_vertical:` apply `\vspace{-\baselineskip}` and set `\topsep=0pt` for the “first level” of the nested `enumext` environment.

```

base-fix
\__enumext_nested_base_line_fix:
975 \keys_define:nn { enumext / level-1 }
976   {
977     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
978     base-fix .initial:n = false,
979     base-fix .value_forbidden:n = true,
980   }

```

The function `\__enumext_nested_base_line_fix:` passed to the `\__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.39) will be responsible for applying the *baseline correction* and adjusting the *(keys)* for the `enumext` environment and the `\printkeyans` with *starred argument* ‘\*’ (§13.47).

We will first implement the function code from the user side of the `base-fix` key, that is, only the user knows when it is necessary to apply it within the document in which case the variable `\l__enumext_print_keyans_star_bool` set by the `\printkeyans` command is false and the variable `\l__enumext_base_line_fix_bool` is true.

```

981 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
982   {
983     \bool_lazy_all:nT
984       {
985         { \bool_if_p:N \l__enumext_starred_first_bool }
986         { \bool_if_p:N \l__enumext_base_line_fix_bool }
987         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
988       }
989     {
990       \mode_leave_vertical:
991       \vspace { -\dim_eval:n { \baselineskip + \parsep } }
992     }

```

When we are running the `\printkeyans` command with the *starred argument* ‘\*’ the variable `\l__enumext_print_keyans_star_bool` is true and we can run a simplified version of `\vspace` using `\skip_vertical:n`.

```

993     \bool_lazy_and:nnT
994       { \bool_if_p:N \l__enumext_starred_first_bool }
995       { \bool_if_p:N \l__enumext_print_keyans_star_bool }
996     {

```

```

997     \mode_leave_vertical:
998     \skip_vertical:n { -\baselineskip }
999     \skip_vertical:N \c_zero_skip
1000 }

```

Finally we set the values of the keys `topsep`, `above` and `above*` for the “first level” of `enumext` environment equal to `0pt` and set the variable `\l__enumext_base_line_fix_bool` to false.

```

1001 \keys_set:nn { enumext / level-1 }
1002 {
1003     topsep = 0pt, above = 0pt, above* = 0pt,
1004 }
1005 \bool_set_false:N \l__enumext_base_line_fix_bool
1006 }

```

(End of definition for `base-fix` and `\l__enumext_nested_base_line_fix:`.)

### 13.18 Setting keys for horizontal spaces

Define and set `itemindent`, `rightmargin`, `listparindent`, `list-offset` and `list-indent` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1007 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1008 {
1009     \keys_define:nn { enumext / #1 }
1010     {
1011         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1012         itemindent .value_required:n = true,
1013         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1014         rightmargin .value_required:n = true,
1015         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1016         listparindent .value_required:n = true,
1017         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1018         list-offset .value_required:n = true,
1019         list-indent .code:n =
1020             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1021             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1022         list-indent .value_required:n = true,
1023     }
1024 }
1025 \clist_map_inline:nn
1026 {
1027     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1028 }
1029 { \l__enumext_tmp:nn #1 }

```

(End of definition for `itemindent` and others.)

For `enumext*` and `keyans*` environments the situation is a bit different, the `list-indent` key behaves like the `list-offset` key.

```

1030 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1031 {
1032     \keys_define:nn { enumext / #1 }
1033     {
1034         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1035         itemindent .value_required:n = true,
1036         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1037         rightmargin .value_required:n = true,
1038         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1039         listparindent .value_required:n = true,
1040         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1041         list-offset .value_required:n = true,
1042         list-indent .meta:n = { list-offset = ##1 },
1043         list-indent .value_required:n = true,
1044     }
1045 }
1046 \clist_map_inline:nn
1047 {
1048     {enumext*}{vii}, {keyans*}{viii}
1049 }
1050 { \l__enumext_tmp:nn #1 }

```

### 13.18.1 Functions for setting the fake itemindent

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `\opt`. Here I will need to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

1051 \cs_set_protected:Nn \__enumext_fake_item_indent:
1052 {
1053   \dim_compare:nNnT
1054     { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1055     >
1056     { \c_zero_dim }
1057   {
1058     \tl_set:ce { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
1059     {
1060       \exp_not:N \mode_leave_vertical:
1061       \exp_not:n { \skip_horizontal:n }
1062       { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1063       \exp_not:N \ignorespaces
1064     }
1065   }
1066 }
1067 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1068 {
1069   \dim_compare:nNnT
1070     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1071     {
1072       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1073       {
1074         \exp_not:N \mode_leave_vertical:
1075         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1076         \exp_not:N \ignorespaces
1077       }
1078     }
1079 }
1080 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1081 {
1082   \dim_compare:nNnT
1083     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1084     {
1085       \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1086       {
1087         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1088         \exp_not:N \ignorespaces
1089       }
1090     }
1091 }
1092 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1093 {
1094   \dim_compare:nNnT
1095     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1096     {
1097       \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1098       {
1099         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1100         \exp_not:N \ignorespaces
1101       }
1102     }
1103 }

```

(End of definition for `\__enumext_fake_item_indent:` and others.)

### 13.19 Setting show-length key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to “true” and calls the function `\__enumext_show_length:nnn` which prints all the values of the “vertical” and “horizontal” parameters calculated and used.

```

1104 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1105 {
1106   \keys_define:nn { enumext / #1 }
1107   {

```

```

1108         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1109         show-length .initial:n = false,
1110     }
1111 }
1112 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for show-length.)

### 13.20 Setting before, after and first keys

Define and set `before`, `before*`, `after` and `first` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

before* 1113 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    1114 {
first    1115     \keys_define:nn { enumext / #1 }
         1116     {
         1117         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
         1118         before .value_required:n = true,
         1119         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
         1120         before* .value_required:n = true,
         1121         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
         1122         after .value_required:n = true,
         1123         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
         1124         first .value_required:n = true,
         1125     }
         1126 }
1127 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for before and others.)

#### 13.20.1 Functions for before, after and first keys in enumext

The function `\__enumext_before_args_exec:` executes the `{⟨code⟩}` set by the `before*` key “before” the `enumext` environment is started. The `{⟨code⟩}` is executed “without” knowing any definition of the `{⟨arg two⟩}` of the list: `{⟨code⟩}\list{⟨arg one⟩}{⟨arg two⟩}`.

```

1128 \cs_new_protected:Nn \__enumext_before_args_exec:
1129 {
1130     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1131 }

```

The function `\__enumext_before_keys_exec:` executes the `{⟨code⟩}` set by the `before` key “before” the `enumext` environment is started in *second argument* of the list. The `{⟨code⟩}` is executed “knowing” all definition and values provides by `⟨keys⟩: \list{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}`

```

1132 \cs_new_protected:Nn \__enumext_before_keys_exec:
1133 {
1134     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1135 }

```

The function `\__enumext_after_stop_list:` executes the `{⟨code⟩}` set by the `after` key “after” the `enumext` environment has finished: `\endlist{⟨code⟩}`.

```

1136 \cs_new_protected:Nn \__enumext_after_stop_list:
1137 {
1138     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1139 }

```

The function `\__enumext_after_args_exec:` executes the `{⟨code⟩}` set by the `first` key after the end of the second argument of the list defining the `enumext` environment, just before the first occurrence of `\item: \list{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}\item.`

```

1140 \cs_new_protected:Nn \__enumext_after_args_exec:
1141 {
1142     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1143 }

```

(End of definition for \\_\_enumext\_before\_args\_exec: and others.)

#### 13.20.2 Functions for before, after and first keys in keyans

Same implementation as the one used in the `enumext` environment.

```

\__enumext_before_args_exec_v: 1144 \cs_new_protected:Nn \__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v: 1145 {
\__enumext_after_stop_list_v: 1146     \tl_use:N \l__enumext_before_starred_key_v_tl
\__enumext_after_args_exec_v: 1147 }
1148 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1149 {

```



```

1150     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1151   }
1152   \cs_new_protected:Nn \__enumext_after_stop_list_v:
1153   {
1154     \tl_use:N \l__enumext_after_stop_list_v_tl
1155   }
1156   \cs_new_protected:Nn \__enumext_after_args_exec_v:
1157   {
1158     \tl_use:N \l__enumext_after_list_args_v_tl
1159   }

```

(End of definition for `\__enumext_before_args_exec_v:` and others.)

### 13.20.3 Functions for before, after and first keys in enumext\* and keyans\*

Same implementation as the one used in the `enumext` environment.

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
1160 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1161 {
1162   \tl_use:N \l__enumext_before_starred_key_vii_tl
1163 }
1164 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1165 {
1166   \tl_use:N \l__enumext_before_starred_key_viii_tl
1167 }
1168 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1169 {
1170   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1171 }
1172 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1173 {
1174   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1175 }
1176 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1177 {
1178   \tl_use:N \l__enumext_after_stop_list_vii_tl
1179 }
1180 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1181 {
1182   \tl_use:N \l__enumext_after_stop_list_viii_tl
1183 }
1184 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1185 {
1186   \tl_use:N \l__enumext_after_list_args_vii_tl
1187 }
1188 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1189 {
1190   \tl_use:N \l__enumext_after_list_args_viii_tl
1191 }

```

(End of definition for `\__enumext_before_args_exec_vii:` and others.)

### 13.21 Setting keys for multicols and minipage

The default value of the `columns-sep` key is handled by the state of the boolean variable `\l__enumext_columns_sep_X_bool` which is handled in the internal definition of the `enumext` and `keyans` environments. Define and set `mini-env`, `mini-sep`, `columns-sep` and `columns` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1192 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1193 {
1194   \keys_define:nn { enumext / #1 }
1195   {
1196     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1197     mini-env .value_required:n = true,
1198     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1199     mini-sep .initial:n = 0.3333em,
1200     mini-sep .value_required:n = true,
1201     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1202     columns-sep .value_required:n = true,
1203     columns .int_set:c = { l__enumext_columns_#2_int },
1204     columns .initial:n = 1,
1205     columns .value_required:n = true,
1206   }

```

```

1207 }
1208 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

For enumext* and keyans* environments the situation is a bit different, the command \miniright is
not available, so we will add the keys mini-right and mini-right* to implement support for minipage
environment.

1209 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1210 {
1211   \keys_define:nn { enumext / #1 }
1212   {
1213     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1214     mini-right .value_required:n = true,
1215     mini-right* .code:n = {
1216       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1217       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1218     },
1219     mini-right* .value_required:n = true,
1220   }
1221 }
1222 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for mini-env and others.)

### 13.22 Adjustment of vertical spaces for multicol

When nesting a “list environment” inside the `multicol` environment, the values of the “vertical spaces” are lost, basically the `multicol` environment takes control over them. Graphically it can be seen like in the figure 7.

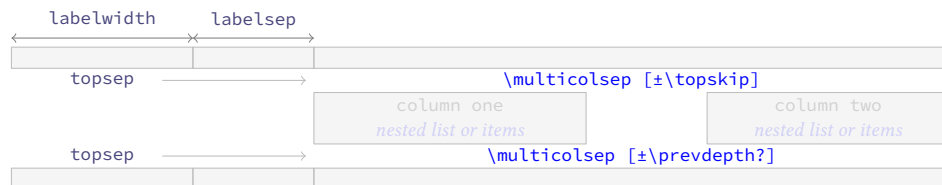


Figure 7: Representation of the vertical space in `multicol` for a nested level.

To keep the desired spaces *above* and *below* in the “list environment” (`\topsep` + `[\partopsep]`) it is necessary to “adjust” the spaces added by the `multicol` environment. The most appropriate option in this case is to use a “context sensitive” vertical space with `\addvspace`.

✎ I should make it clear that the implementation here is a “bit questionable”. At first glance doing `\multicolsep=\topsep` seemed right, but the results were not always as expected. An almost *imperceptible* detail is that in some cases the `\itemsep` values are “stretched”, possibly due to the use of `\raggedcolumns` and this affects the lower space when closing the environment, which is “smaller” than expected. My attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

#### 13.22.1 Adjustment of vertical spaces for multicol in enumext

`\__enumext_multi_set_vskip:` The function `\__enumext_multi_set_vskip:` will take care of determining the “adjusted spaces” that we will apply “above” and “below” the `multicol` environment in `enumext`.

We will set the default values taking into account that T<sub>E</sub>X is in *horizontal mode*, then we will make the settings for the *vertical mode* in which `\partopsep` comes into play.

Set the values of `\l__enumext_multicol_above_X_skip` and `\l__enumext_multicol_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

1223 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1224 {
1225   \skip_set:cn { l__enumext_multicol_above_ \__enumext_level: _skip }
1226   {
1227     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1228   }
1229   \skip_set:cn { l__enumext_multicol_below_ \__enumext_level: _skip }
1230   {
1231     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1232   }
1233   \__enumext_add_pre_parsep:
1234 }

```

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

`\__enumext_add_pre_parsep:` The function `\__enumext_add_pre_parsep:` “adjusted” the value of `\l__enumext_multicol_above_X_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces*.

```

1235 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1236 {
1237   \int_case:nn { \l__enumext_level_int }
1238   {
1239     { 2 }{
1240       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1241       {
1242         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1243         {
1244           \l__enumext_parsep_i_skip
1245         }
1246       }
1247     }
1248     { 3 }{
1249       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1250       {
1251         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1252         {
1253           \l__enumext_parsep_ii_skip
1254         }
1255       }
1256     }
1257     { 4 }{
1258       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1259       {
1260         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1261         {
1262           \l__enumext_parsep_iii_skip
1263         }
1264       }
1265     }
1266   }
1267 }

```

(End of definition for `\__enumext_add_pre_parsep:`)

`\__enumext_multi_addvspace:` The function `\__enumext_multi_addvspace:` will apply the spaces set using `\addvspace` “above” the `multicols` environment in `enumext`, taking into account whether  $\TeX$  is in *horizontal mode* or *vertical mode*.

```

1268 \cs_new_protected:Nn \__enumext_multi_addvspace:
1269 {
1270   \__enumext_multi_set_vskip:
1271   \mode_if_vertical:T
1272   {
1273     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1274     {
1275       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1276     }
1277     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1278     {
1279       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1280     }
1281   }
1282   \par\nopagebreak
1283   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1284 }

```

(End of definition for `\__enumext_multi_addvspace:`)

### 13.22.2 Adjustment of vertical spaces for multicols in keyans

`\__enumext_keyans_multi_set_vskip:` The function `\__enumext_keyans_multi_set_vskip:` will take care of determining the “adjusted spaces” that we will apply “above” and “below” the `multicols` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

`\__enumext_keyans_multi_addvspace:`

```

1285 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1286 {
1287   \skip_set:Nn \l__enumext_multicols_above_v_skip
1288   {
1289     \l__enumext_topsep_v_skip
1290   }
1291   \skip_set:Nn \l__enumext_multicols_below_v_skip
1292   {

```

```

1293     \l__enumext_topsep_v_skip
1294   }
1295 }
1296 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1297 {
1298   \__enumext_keyans_multi_set_vskip:
1299   \mode_if_vertical:T
1300   {
1301     \skip_add:Nn \l__enumext_multicols_above_v_skip
1302     {
1303       \skip_use:N \l__enumext_partopsep_v_skip
1304     }
1305     \skip_add:Nn \l__enumext_multicols_below_v_skip
1306     {
1307       \skip_use:N \l__enumext_partopsep_v_skip
1308     }
1309   }
1310   \par\nopagebreak
1311   \addvspace{ \l__enumext_multicols_above_v_skip }
1312 }

```

(End of definition for `\__enumext_keyans_multi_set_vskip:` and `\__enumext_keyans_multi_addvspace:`.)

### 13.23 Adjustment of vertical spaces for minipage

When nesting a “list environment” within the `minipage` environment, the values of the “vertical spaces” are lost. Graphically it can be seen like in the figure 8.

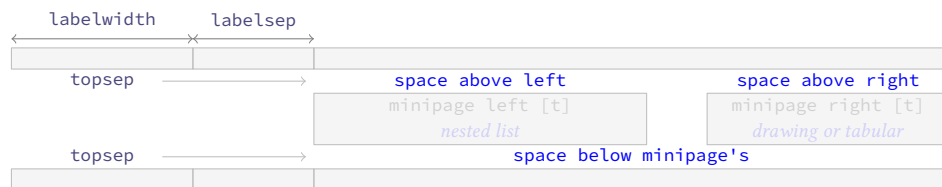


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

Since we want to keep the “left” and “right” environments “aligned on top”, preserving the `\baselineskip` and keep the desired “spaces” (`\topsep` + `[\partopsep]`) it is necessary to “adjust” the “vertical spaces” for `minipage` environments.

Here there are several complications that we must circumvent, the `minipage` environment eliminates the “top” spaces, the `multicols` environment can be nested in the `minipage` environment, the “top” and “bottom” spaces are affected when `topsep=0pt` and to this is added the `\partopsep` parameter that comes into action according to whether  $\TeX$  is in *horizontal mode* or *vertical mode*. Depending on these cases, small adjustments must be made using `\vspace` and `\addvspace` to obtain the “desired vertical spacing”.

Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (glue) produced by the `minipage` environment is quite complicated, even more if `multicols` is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug`[14] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

#### 13.23.1 Adjustment of vertical spaces for minipage in enumext

```

\__enumext_minipage_set_skip:
\__enumext_minipage_add_space:

```

The function `\__enumext_minipage_set_skip:` will take care of determining the “adjust” spaces that we will apply “above” and “below” the `__enumext_mini_page` environment in `enumext`.

First we will set the value of `\l__enumext_minipage_right_skip` equal to `\topsep`, then we will see if  $\TeX$  is in *vertical mode* and we will add `\partopsep`, followed by that we set the value of `\l__enumext_minipage_after_skip`.

```

1313 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1314 {
1315   \skip_set:Nn \l__enumext_minipage_right_skip
1316   {
1317     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1318   }
1319   \mode_if_vertical:T
1320   {
1321     \skip_add:Nn \l__enumext_minipage_right_skip
1322     {
1323       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1324     }
1325   }
1326   \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip

```

We will adjust the values `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip` and call the function `\__enumext_pre_itemsep_skip:`.

```

1327 \skip_set_eq:cN
1328 { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1329 \skip_set_eq:cN
1330 { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1331 \__enumext_pre_itemsep_skip:

```

If the environment `multicols` is active, we set `\topskip=0pt` and then we make `\multicolsep` have the same value as `\l__enumext_multicols_above_X_skip`.

```

1332 \int_compare:nNtT
1333 { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1334 {
1335   \skip_zero:N \topskip
1336   \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1337 }
1338 }

```

The function `\__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `\__enumext_mini_page` environment, taking into account whether `TeX` is in *horizontal mode* or *vertical mode*. Here we use the plain `TeX` macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1339 \cs_new_protected:Nn \__enumext_minipage_add_space:
1340 {
1341   \__enumext_minipage_set_skip:
1342   \__enumext_unskip_unkern:
1343   \mode_if_vertical:TF
1344   {
1345     \nopagebreak\nointerlineskip
1346   }
1347   {
1348     \par\nopagebreak\nointerlineskip
1349     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1350   }
1351   \int_compare:nNtTF
1352   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1353   {
1354     \addvspace{ 0.445\box_ht:N \strutbox }
1355   }
1356   {
1357     \addvspace{ 0.250\box_ht:N \strutbox }
1358   }
1359 }

```

(End of definition for `\__enumext_minipage_set_skip:` and `\__enumext_minipage_add_space:`.)

`\__enumext_pre_itemsep_skip:` The function `\__enumext_pre_itemsep_skip:` will adjust the spaces below the environment `minipage` and the environment `multicols` if it is nested in it, taking into account the value of `\itemsep` from the previous level.

```

1360 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1361 {
1362   \int_case:nn { \l__enumext_level_int }
1363   {
1364     { 2 }{
1365       \skip_if_eq:nnTF
1366       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1367       {
1368         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1369         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1370       }
1371       {
1372         \dim_compare:nNtT
1373         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1374         {
1375           \skip_sub:Nn
1376           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1377           \skip_sub:Nn
1378           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1379           \skip_add:Nn
1380           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }

```

```

1381         \skip_add:Nn
1382         \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1383     }
1384     \dim_compare:nNnT
1385     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1386     {
1387         \skip_set:Nn \l__enumext_minipage_temp_skip
1388         {
1389             \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1390         }
1391         \skip_sub:Nn
1392         \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1393         \skip_sub:Nn
1394         \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1395         \skip_add:Nn
1396         \l__enumext_minipage_after_skip
1397         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1398         \skip_add:Nn
1399         \l__enumext_multicols_below_ii_skip
1400         { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1401     }
1402 }
1403 }
1404 { 3 }{
1405     \skip_if_eq:nNnTF
1406     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1407     {
1408         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1409         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1410     }
1411     {
1412         \dim_compare:nNnT
1413         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1414         {
1415             \skip_sub:Nn
1416             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1417             \skip_sub:Nn
1418             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1419             \skip_add:Nn
1420             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1421             \skip_add:Nn
1422             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1423         }
1424         \dim_compare:nNnT
1425         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1426         {
1427             \skip_set:Nn \l__enumext_minipage_temp_skip
1428             {
1429                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1430             }
1431             \skip_sub:Nn
1432             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1433             \skip_sub:Nn
1434             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1435             \skip_add:Nn
1436             \l__enumext_minipage_after_skip
1437             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1438             \skip_add:Nn
1439             \l__enumext_multicols_below_iii_skip
1440             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1441         }
1442     }
1443 }
1444 { 4 }{
1445     \skip_if_eq:nNnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1446     {
1447         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1448         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1449     }
1450     {
1451         \dim_compare:nNnT

```



```

1452         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1453         {
1454             \skip_sub:Nn
1455               \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1456             \skip_sub:Nn
1457               \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1458             \skip_add:Nn
1459               \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1460             \skip_add:Nn
1461               \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1462         }
1463     \dim_compare:nNtT
1464     { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1465     {
1466         \skip_set:Nn \l__enumext_minipage_temp_skip
1467         {
1468             \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1469         }
1470         \skip_sub:Nn
1471           \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1472         \skip_sub:Nn
1473           \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1474         \skip_add:Nn
1475           \l__enumext_minipage_after_skip
1476           { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1477         \skip_add:Nn
1478           \l__enumext_multicols_below_iv_skip
1479           { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1480     }
1481 }
1482 }
1483 }
1484 }

```

(End of definition for `\__enumext_pre_itemsep_skip:`)

### 13.23.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

The function `\__enumext_keyans_mini_set_vskip:` will take care of determining the “adjusted” spaces that we will apply “*above*” and “*below*” the `\__enumext_mini_page` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

```

1485 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1486 {
1487     \skip_zero:N \l__enumext_minipage_after_skip
1488     \skip_zero:N \l__enumext_minipage_left_skip
1489     \skip_zero:N \l__enumext_minipage_right_skip
1490     \skip_set:Nn \l__enumext_minipage_right_skip
1491     {
1492         \l__enumext_topsep_v_skip
1493     }
1494     \mode_if_vertical:T
1495     {
1496         \skip_add:Nn \l__enumext_minipage_right_skip
1497         {
1498             \l__enumext_partopsep_v_skip
1499         }
1500     }
1501     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1502     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1503     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1504     \__enumext_keyans_pre_itemsep_skip:
1505     \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
1506     {
1507         \skip_zero:N \topskip
1508         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1509     }
1510 }
1511 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1512 {
1513     \__enumext_keyans_minipage_set_skip:
1514     \__enumext_unskip_unkern:
1515     \mode_if_vertical:TF

```

```

1516     {
1517         \nopagebreak\nointerlineskip
1518     }
1519     {
1520         \par\nopagebreak\nointerlineskip
1521         \skip_zero:N \l__enumext_partopsep_v_skip
1522     }
1523     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1524     {
1525         \addvspace{ 0.445\box_ht:N \strutbox }
1526     }
1527     {
1528         \addvspace{ 0.250\box_ht:N \strutbox }
1529     }
1530 }
1531 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1532 {
1533     \skip_if_eq:nnTF
1534     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1535     {
1536         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1537         \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1538     }
1539     {
1540         \dim_compare:nNnT
1541         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1542         {
1543             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1544             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1545             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1546             \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1547         }
1548         \dim_compare:nNnT
1549         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1550         {
1551             \skip_set:Nn \l__enumext_minipage_temp_skip
1552             {
1553                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1554             }
1555             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1556             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1557             \skip_add:Nn \l__enumext_minipage_after_skip
1558             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1559             \skip_add:Nn \l__enumext_multicols_below_v_skip
1560             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1561         }
1562     }
1563 }

```

(End of definition for `\__enumext_keyans_minipage_set_skip:`, `\__enumext_keyans_minipage_add_space:`, and `\__enumext_keyans_pre_itemsep_skip:`.)

### 13.23.3 Adjustment of vertical spaces for minipage in enumext\* and keyans\*

`\__enumext_mini_set_vskip_vii:`  
`\__enumext_mini_set_vskip_viii:`

The functions `\__enumext_mini_set_vskip_vii:` and `\__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `\__enumext_mini_page` environment in `enumext*` and `keyans*`.

```

1564 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1565 {
1566     \skip_zero_new:N \l__enumext_minipage_left_skip
1567     \skip_gzero_new:N \g__enumext_minipage_right_skip
1568     \skip_gzero_new:N \g__enumext_minipage_after_skip
1569     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1570     {
1571         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1572         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1573     }
1574     {
1575         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1576         \skip_gset:Nn \g__enumext_minipage_right_skip
1577         {
1578             \l__enumext_topsep_vii_skip

```

```

1579     }
1580     \skip_gset:Nn \g__enumext_minipage_after_skip
1581     {
1582         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1583     }
1584 }
1585 }
1586 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1587 {
1588     \skip_zero_new:N \l__enumext_minipage_after_skip
1589     \skip_zero_new:N \l__enumext_minipage_left_skip
1590     \skip_zero_new:N \l__enumext_minipage_right_skip
1591     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1592     {
1593         \skip_set:Nn \l__enumext_minipage_left_skip
1594         {
1595             0.5\box_dp:N \strutbox
1596         }
1597         \skip_set:Nn \l__enumext_minipage_right_skip
1598         {
1599             \l__enumext_partopsep_viii_skip
1600         }
1601         \skip_set:Nn \l__enumext_minipage_after_skip
1602         {
1603             1.6\box_dp:N \strutbox
1604         }
1605     }
1606 }
1607 \skip_set:Nn \l__enumext_minipage_left_skip
1608 {
1609     0.5875\box_dp:N \strutbox
1610 }
1611 \skip_set:Nn \l__enumext_minipage_right_skip
1612 {
1613     \l__enumext_topsep_viii_skip
1614 }
1615 \skip_set:Nn \l__enumext_minipage_after_skip
1616 {
1617     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1618 }
1619 }
1620 }

```

(End of definition for `\__enumext_mini_set_vskip_vii:` and `\__enumext_mini_set_vskip_viii:`)

`\__enumext_mini_addvspace_vii:`  
`\__enumext_mini_addvspace_viii:`

The functions `\__enumext_mini_addvspace_vii:` and `\__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `\__enumext_mini_page` environment on the *left side* when the *mini-right* key is active in the `enumext*` and `keyans*` environments.

Here we will NOT take into account whether TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1621 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1622 {
1623     \__enumext_mini_set_vskip_vii:
1624     \par\nopagebreak
1625     \addvspace { \l__enumext_minipage_left_skip }
1626 }
1627 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1628 {
1629     \__enumext_mini_set_vskip_viii:
1630     \par\nopagebreak
1631     \addvspace { \l__enumext_minipage_left_skip }
1632 }

```

(End of definition for `\__enumext_mini_addvspace_vii:` and `\__enumext_mini_addvspace_viii:`)

### 13.23.4 The command `\miniright`

The command `\miniright` will close the `\__enumext_mini_page` environment on the “left side”, open the `\__enumext_mini_page` environment on the “right side” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “right side” environment. The *starred argument* ‘*\**’ inhibits the use of `\centering` command i.e. the usual L<sup>A</sup>T<sub>E</sub>X justification is maintained in the `\__enumext_mini_page` on the “right side”.

`\miniright` First we will perform some checks to prevent the command from being executed outside the `enumext` environment or somewhere inappropriate then we will call the internal functions to execute it in the `enumext` and `keyans` environments.

```

1633 \NewDocumentCommand \miniright { s }
1634 {
1635   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1636   {
1637     \msg_error:nnn { enumext } { wrong-miniright-place }
1638   }
1639   % outside
1640   \bool_lazy_and:nnT
1641   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
1642   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
1643   {
1644     \msg_error:nnn { enumext } { wrong-miniright-place }
1645   }
1646   % starred env
1647   \bool_if:NT \__enumext_starred_bool
1648   {
1649     \msg_error:nnn { enumext } { wrong-miniright-starred }
1650   }
1651   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1652   {
1653     \__enumext_keyans_mini_right_cmd:n {#1}
1654   }
1655   { \__enumext_mini_right_cmd:n {#1} }
1656 }

```

(End of definition for `\miniright`. This function is documented on page 11.)

`\__enumext_mini_right_cmd:n`

The function `\__enumext_mini_right_cmd:n` takes as argument the *starred* ‘\*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\__enumext_minipage_right_X_dim`, if so we close the `multicols` environment with the `\__enumext_mini_page` environment on the “left side”, then we open the `\__enumext_mini_page` environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the `\centering` command when the *starred argument* ‘\*’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1657 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1658 {
1659   \dim_compare:nNtF
1660   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1661   {
1662     \__enumext_multicols_stop:
1663     \int_compare:nNtF
1664     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } = { 1 }
1665     {
1666       \par\addvspace{ \__enumext_minipage_after_skip }
1667     }
1668     \end__enumext_mini_page
1669     \hfill
1670     \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1671     \par\nointerlineskip
1672     \addvspace { \__enumext_minipage_right_skip }
1673     \bool_if:nF {#1}
1674     {
1675       \centering
1676     }
1677     \int_gzero:N \g__enumext_minipage_stat_int
1678   }
1679   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1680   % paranoia
1681   \RenewDocumentCommand \miniright { s }
1682   {
1683     \msg_error:nn { enumext } { many-miniright-used }
1684   }
1685 }

```

(End of definition for `\__enumext_mini_right_cmd:n`.)

`\__enumext_keyans_mini_right_cmd:n`

The function `\__enumext_keyans_mini_right_cmd:n` takes as argument the *starred* ‘\*’ of the `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `\__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1686 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1687 {
1688   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1689   {
1690     \__enumext_keyans_multicols_stop:
1691     \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1692     {
1693       \par\addvspace{ \l__enumext_minipage_after_skip }
1694     }
1695     \end__enumext_mini_page
1696     \hfill
1697     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1698     \par\nointerlineskip
1699     \addvspace { \l__enumext_minipage_right_skip }
1700     \bool_if:nF { #1 }
1701     {
1702       \centering
1703     }
1704     \int_gzero:N \g__enumext_minipage_stat_int
1705   }
1706   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1707 % paranoia
1708 \RenewDocumentCommand \miniright { s }
1709 {
1710   \msg_error:nn { enumext } { many-miniright-used }
1711 }
1712 }

```

(End of definition for \\_\_enumext\_keyans\_mini\_right\_cmd:n.)

### 13.24 Setting above and below keys

While having controlled the *vertical spaces* within the `enumext` and `keyans` environments when using the `columns` or `mini-env` keys, sometimes the “vertical spaces above” or “vertical spaces below” the environments are not as expected and it is necessary to be able to apply a “fine correction” to these. As I have not been able to correct these *glitches*, the best option is to leave a couple of *keys* dedicated to this purpose, in this case it is best to use `\vspace` or `\vspace*` when convenient.

Define `above`, `above*`, `below` and `below*` keys for `enumext` and `keyans` environments.

```

1713 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1714 {
1715   \keys_define:nn { enumext / #1 }
1716   {
1717     above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1718     above .value_required:n = true,
1719     above* .code:n      = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1720                       \keys_set:nn { enumext / #1 } { above = {##1} },
1721     above* .value_required:n = true,
1722     below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1723     below .value_required:n = true,
1724     below* .code:n      = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1725                       \keys_set:nn { enumext / #1 } { below = {##1} },
1726     below* .value_required:n = true,
1727   }
1728 }
1729 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `above` and others.)

#### 13.24.1 Functions for above and below keys in enumext

`\__enumext_vspace_above:` The function `\__enumext_vspace_above:` apply the *vertical space above* the `enumext` environment set by the `above*` and `above` keys.

```

1730 \cs_new_protected:Nn \__enumext_vspace_above:
1731 {
1732   \skip_if_eq:nnF
1733   { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1734   {
1735     \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1736     {
1737       \vspace*{ \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1738     }
1739     {

```

```

1740         \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1741     }
1742 }
1743 }

```

(End of definition for \\_\_enumext\_vspace\_above:.)

\\_\_enumext\_vspace\_below: The function \\_\_enumext\_vspace\_below: apply the *vertical space below* the `enumext` environment set by the *below\** and *below* keys.

```

1744 \cs_new_protected:Nn \__enumext_vspace_below:
1745 {
1746     \skip_if_eq:nnF
1747     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1748     {
1749         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1750         {
1751             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1752         }
1753         {
1754             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1755         }
1756     }
1757 }

```

(End of definition for \\_\_enumext\_vspace\_below:.)

### 13.24.2 Functions for above and below keys in keyans

\\_\_enumext\_vspace\_above\_v: The function \\_\_enumext\_vspace\_above\_v: apply the *vertical space above* the `keyans` environment set by the *above* and *above\** keys.

```

1758 \cs_new_protected:Nn \__enumext_vspace_above_v:
1759 {
1760     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1761     {
1762         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1763         {
1764             \vspace*{ \l__enumext_vspace_above_v_skip }
1765         }
1766         { \vspace { \l__enumext_vspace_above_v_skip } }
1767     }
1768 }

```

(End of definition for \\_\_enumext\_vspace\_above\_v:.)

\\_\_enumext\_vspace\_below\_v: The function \\_\_enumext\_vspace\_below\_v: apply the *vertical space below* the `keyans` environment set by the *below\** and *below* keys.

```

1769 \cs_new_protected:Nn \__enumext_vspace_below_v:
1770 {
1771     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1772     {
1773         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1774         {
1775             \vspace*{ \l__enumext_vspace_below_v_skip }
1776         }
1777         { \vspace { \l__enumext_vspace_below_v_skip } }
1778     }
1779 }

```

(End of definition for \\_\_enumext\_vspace\_below\_v:.)

### 13.24.3 Functions for above and below keys in enumext\* keyans\*

\\_\_enumext\_vspace\_above\_vii: The functions \\_\_enumext\_vspace\_above\_vii: and \\_\_enumext\_vspace\_above\_viii: apply the *vertical space above* the `enumext*` and `keyans*` environments set by the *above* and *above\** keys.

\\_\_enumext\_vspace\_above\_viii:

```

1780 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1781 {
1782     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1783     {
1784         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1785         {
1786             \vspace*{ \l__enumext_vspace_above_vii_skip }
1787         }
1788         { \vspace { \l__enumext_vspace_above_vii_skip } }

```



```

1789     }
1790   }
1791   \cs_new_protected:Nn \__enumext_vspace_above_viii:
1792   {
1793     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1794     {
1795       \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1796       {
1797         \vspace*{ \l__enumext_vspace_above_viii_skip }
1798       }
1799       { \vspace { \l__enumext_vspace_above_viii_skip } }
1800     }
1801   }

```

(End of definition for \\_\_enumext\_vspace\_above\_vii: and \\_\_enumext\_vspace\_above\_viii:.)

```

\__enumext_vspace_below_vii:
\__enumext_vspace_below_viii:

```

The functions \\_\_enumext\_vspace\_below\_vii: and \\_\_enumext\_vspace\_below\_viii: apply the *vertical space below* the **enumext\*** and **keyans\*** environments set by the **below\*** and **below** keys.

```

1802 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1803 {
1804   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1805   {
1806     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1807     {
1808       \vspace*{ \l__enumext_vspace_below_vii_skip }
1809     }
1810     { \vspace { \l__enumext_vspace_below_vii_skip } }
1811   }
1812 }
1813 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1814 {
1815   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1816   {
1817     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1818     {
1819       \vspace*{ \l__enumext_vspace_below_viii_skip }
1820     }
1821     { \vspace { \l__enumext_vspace_below_viii_skip } }
1822   }
1823 }

```

(End of definition for \\_\_enumext\_vspace\_below\_vii: and \\_\_enumext\_vspace\_below\_viii:.)

### 13.25 Setting series, resume and resume\* keys

The **series** key is responsible for the whole process of the **resume** and **resume\*** keys. The idea behind this is to be able to absorb the *⟨keys⟩* passed to the *optional argument* of the “first level” of the environments **enumext** and **enumext\***, but, discarding some specific *⟨keys⟩*. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in **chat-TeX-SX**

```

series  We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1824 \cs_set_protected:Npn \__enumext_tmp:n #1
1825 {
1826   \keys_define:nn { enumext / #1 }
1827   {
1828     series .str_set:N = \l__enumext_series_str,
1829     series .value_required:n = true,
1830     resume .code:n = \__enumext_resume_series:n {##1},
1831     resume* .code:n = \__enumext_resume_starred:,
1832     resume* .value_forbidden:n = true,
1833   }
1834 }
1835 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for series, resume, and resume\*.)

### 13.25.1 Internal functions for series key

```

__enumext_filter_series:n
  __enumext_filter_series_key:n
  __enumext_filter_series_pair:nn

```

The function `__enumext_filter_series:n` will be in charge of filtering the *⟨keys⟩* we want to store where *{#1}* represents the *optional argument* passed to the environment.

```

1836 \cs_new:Npn __enumext_filter_series:n #1
1837 {
1838   \use:e
1839   {
1840     \keyval_parse:NNn
1841     __enumext_filter_series_key:n
1842     __enumext_filter_series_pair:nn {#1}
1843   }
1844 }

```

The function `__enumext_filter_series_key:n` will be responsible for filtering the *⟨keys⟩* that are passed “without value” by excluding the `resume`, `resume*` and `base-fix` keys.

```

1845 \cs_new:Npn __enumext_filter_series_key:n #1
1846 {
1847   \str_case:nnF {#1}
1848   {
1849     { resume } {} { resume* } {} { base-fix } {}
1850   }
1851   { , { \exp_not:n {#1} } }
1852 }

```

The function `__enumext_filter_series_pair:nn` will be responsible for filtering the *⟨keys⟩* that are passed “with value” by excluding the `series`, `resume`, `start`, `start*`, `save-ans` and `save-key` keys.

```

1853 \cs_new:Npn __enumext_filter_series_pair:nn #1#2
1854 {
1855   \str_case:nnF {#1}
1856   {
1857     { series } {} { resume } {} { start } {}
1858     { start* } {} { save-ans } {} { save-key } {}
1859   }
1860   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1861 }

```

(End of definition for `__enumext_filter_series:n`, `__enumext_filter_series_key:n`, and `__enumext_filter_series_pair:nn`.)

```

__enumext_parse_series:n
__enumext_resume_last:n

```

The function `__enumext_parse_series:n` will be responsible for storing the filtered *⟨keys⟩* in the global variable `g__enumext_series_⟨series name⟩_tl` along with the creation of the integer variable `g__enumext_series_⟨series name⟩_int` when the key is passed as an argument; otherwise, it will check the state of the boolean variable `l__enumext_resume_active_bool` set by the keys `resume` and `resume*` and will call the function `__enumext_resume_last:n`.

🔴 The value of boolean variable `l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *⟨keys⟩*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.39) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.44).

```

1862 \cs_new_protected:Npn __enumext_parse_series:n #1
1863 {
1864   \str_if_empty:NTF \l__enumext_series_str
1865   {
1866     \bool_if:NF \l__enumext_resume_active_bool
1867     {
1868       __enumext_resume_last:n {#1}
1869     }
1870   }
1871   {
1872     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1873     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1874     { __enumext_filter_series:n {#1} }
1875     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1876     {
1877       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1878     }
1879   }
1880 }

```

The function `\__enumext_resume_last:n` will be in charge of saving the filtering (*keys*) when the *series* key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the *enumext* environment and in the variable `\g__enumext_starred_series_tl` for the *enumext\** environment.

```

1881 \cs_new_protected:Npn \__enumext_resume_last:n #1
1882 {
1883   \bool_if:NT \l__enumext_standar_first_bool
1884   {
1885     \tl_gclear:N \g__enumext_standar_series_tl
1886     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1887   }
1888   \bool_if:NT \l__enumext_starred_first_bool
1889   {
1890     \tl_gclear:N \g__enumext_starred_series_tl
1891     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1892   }
1893 }

```

(End of definition for `\__enumext_parse_series:n` and `\__enumext_resume_last:n`)

### 13.25.2 Internal function to save counter value

`\__enumext_resume_save_counter:`

The `\__enumext_resume_save_counter:` function will save the last counter value to `\g__enumext_series_⟨series name⟩_int` if the *series*={⟨series name⟩} key has been passed, to `\g__enumext_resume_⟨series name⟩_int` if it has passed the key *resume without value* and the key *series* is not active, in `\g__enumext_series_⟨series name⟩_int` if the key *resume*={⟨series name⟩} has been passed and in `\g__enumext_series_⟨store name⟩_int` if the key has been passed *save-ans*={⟨store name⟩}.

🔗 The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same {⟨series name⟩} but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute *series*={⟨series name⟩} and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use *resume*={⟨series name⟩}. This function is passed to the *enumext* environment definition (§13.39) and the *enumext\** environment definition (§13.44).

```

1894 \cs_new_protected:Npn \__enumext_resume_save_counter:
1895 {
1896   \bool_if:NT \g__enumext_standar_bool
1897   {
1898     \tl_if_empty:NF \l__enumext_series_str
1899     {
1900       \int_gset_eq:cN
1901       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1902     }
1903     \tl_if_empty:NTF \l__enumext_resume_name_tl
1904     {
1905       \str_if_empty:NT \l__enumext_series_str
1906       {
1907         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1908       }
1909     }
1910     {
1911       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1912       {
1913         \int_gset_eq:cN
1914         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1915       }
1916     }
1917     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1918     {
1919       \int_gset_eq:cN
1920       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1921     }
1922   }
1923   \bool_if:NT \g__enumext_starred_bool
1924   {
1925     \tl_if_empty:NF \l__enumext_series_str
1926     {
1927       \int_gset_eq:cN
1928       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1929     }
1930     \tl_if_empty:NTF \l__enumext_resume_name_tl
1931     {
1932       \str_if_empty:NT \l__enumext_series_str
1933       {

```

```

1934         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1935     }
1936 }
1937 {
1938     \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1939     {
1940         \int_gset_eq:cN
1941         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1942     }
1943 }
1944 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1945 {
1946     \int_gset_eq:cN
1947     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1948 }
1949 }
1950 }

```

(End of definition for \\_\_enumext\_resume\_save\_counter:.)

### 13.25.3 Internal functions for resume key

\\_\_enumext\_resume\_series:n

The function \\_\_enumext\_resume\_series:n will handle the argument passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function \\_\_enumext\_resume\_counter: is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is not present, if the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key, otherwise it will verify that the `\g__enumext_series_⟨series name⟩_tl` variable set by the `series` key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```

1951 \cs_new_protected:Npn \__enumext_resume_series:n #1
1952 {
1953     \tl_if_empty:nTF {#1}
1954     {
1955         \__enumext_resume_counter:n { }
1956     }
1957     {
1958         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1959         {
1960             \__enumext_resume_counter:n {#1}
1961             \bool_if:NT \g__enumext_standar_bool
1962             {
1963                 \keys_set:nv { enumext / level-1 }
1964                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1965             }
1966             \bool_if:NT \g__enumext_starred_bool
1967             {
1968                 \keys_set:nv { enumext / enumext* }
1969                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1970             }
1971         }
1972         {
1973             \bool_if:NT \g__enumext_standar_bool
1974             {
1975                 \msg_error:nnn { enumext } { unknown-series } {#1}
1976             }
1977             \bool_if:NT \g__enumext_starred_bool
1978             {
1979                 \msg_error:nnn { enumext } { unknown-series } {#1}
1980             }
1981         }
1982     }
1983 }

```

(End of definition for \\_\_enumext\_resume\_series:n)

\\_\_enumext\_resume\_counter:n

\\_\_enumext\_resume\_counter:

\\_\_enumext\_resume\_counter\_series:

\\_\_enumext\_resume\_counter\_save\_ans:

The function \\_\_enumext\_resume\_counter:n will set the variable \l\_\_enumext\_resume\_active\_bool to true and pass the value of the key `resume` to the variable \l\_\_enumext\_series\_name\_tl which will contain the {⟨series name⟩}. If the variable \l\_\_enumext\_series\_name\_tl is empty, that is, we are passing the key `resume` *without value*, we will execute the function \\_\_enumext\_resume\_counter: otherwise, when we pass `resume={⟨series name⟩}` we will execute the function \\_\_enumext\_resume\_counter\_series:,

finally we will execute the function `\__enumext_resume_counter_save_ans`: which is associated with the key `save-ans`.

```

1984 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1985 {
1986   \bool_set_true:N \l__enumext_resume_active_bool
1987   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1988   \tl_if_empty:NTF \l__enumext_resume_name_tl
1989   {
1990     \__enumext_resume_counter:
1991   }
1992   {
1993     \__enumext_resume_counter_series:
1994   }
1995   \__enumext_resume_counter_save_ans:
1996 }

```

The `\__enumext_resume_counter:` function is executed when the `resume` key is used *without value*, only the counters for the “*first level*” of the environments will be set.

```

1997 \cs_new_protected:Nn \__enumext_resume_counter:
1998 {
1999   \bool_if:NT \g__enumext_standar_bool
2000   {
2001     \int_gincr:N \g__enumext_resume_int
2002     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2003   }
2004   \bool_if:NT \g__enumext_starred_bool
2005   {
2006     \int_gincr:N \g__enumext_resume_vii_int
2007     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2008   }
2009 }

```

The function `\__enumext_resume_counter_series:` will be executed when the `resume={⟨series name⟩}` key is active, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the `series` key.

```

2010 \cs_new_protected:Nn \__enumext_resume_counter_series:
2011 {
2012   \bool_if:NT \g__enumext_standar_bool
2013   {
2014     \int_set:Nn \l__enumext_start_i_int
2015     {
2016       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2017     }
2018   }
2019   \bool_if:NT \g__enumext_starred_bool
2020   {
2021     \int_set:Nn \l__enumext_start_vii_int
2022     {
2023       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2024     }
2025   }
2026 }

```

The function `\__enumext_resume_counter_save_ans:` will be executed when the `save-ans` key is active along with the `resume` key, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the `save-ans` key.

```

2027 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2028 {
2029   \bool_lazy_and:nnT
2030   { \bool_if_p:N \l__enumext_standar_first_bool }
2031   { \bool_if_p:N \l__enumext_store_active_bool }
2032   {
2033     \int_set:Nn \l__enumext_start_i_int
2034     {
2035       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2036     }
2037   }
2038   \bool_lazy_and:nnT
2039   { \bool_if_p:N \l__enumext_starred_first_bool }
2040   { \bool_if_p:N \l__enumext_store_active_bool }
2041   {
2042     \int_set:Nn \l__enumext_start_vii_int

```

```

2043         {
2044             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2045         }
2046     }
2047 }

```

(End of definition for \\_\_enumext\_resume\_counter:n and others.)

### 13.25.4 Internal function for resume\* key

\\_\_enumext\_resume\_starred: The function \\_\_enumext\_resume\_starred: will handle the `resume*` key in the `enumext` and `enumext*` environments. This function will execute the filtered `<keys>` in the last one and will continue with the numbering according to the last execution of the environment `enumext` or `enumext*` in which the keys `resume={<series name>}` or `series={<series name>}` were not active.

```

2048 \cs_new_protected:Nn \__enumext_resume_starred:
2049 {
2050     \bool_if:NT \g__enumext_standar_bool
2051     {
2052         \tl_if_empty:NF \g__enumext_standar_series_tl
2053         {
2054             \__enumext_resume_counter:n { }
2055             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2056         }
2057     }
2058     \bool_if:NT \g__enumext_starred_bool
2059     {
2060         \tl_if_empty:NF \g__enumext_starred_series_tl
2061         {
2062             \__enumext_resume_counter:n { }
2063             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2064         }
2065     }
2066 }

```

(End of definition for \\_\_enumext\_resume\_starred:.)

## 13.26 Setting save-ans, check-ans and no-store keys

The key `save-ans` is directly associated with the keys `check-ans`, `no-store`, `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

### 13.26.1 Setting save-ans key

`save-ans` We define the keys `save-ans` only for the “first level” of `enumext` and `enumext*`.

```

2067 \cs_set_protected:Npn \__enumext_tmp:n #1
2068 {
2069     \keys_define:nn { enumext / #1 }
2070     {
2071         save-ans .code:n = \__enumext_storing_set:n {##1},
2072         save-ans .value_required:n = true,
2073     }
2074 }
2075 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for save-ans.)

### 13.26.2 Internal functions for save-ans key

\\_\_enumext\_start\_save\_ans\_msg: The functions \\_\_enumext\_start\_save\_ans\_msg: and \\_\_enumext\_stop\_save\_ans\_msg: will display in the terminal and .log file the environment in which the `save-ans` key was executed along with the line at the beginning and end of it. The function \\_\_enumext\_start\_save\_ans\_msg: will be passed to \\_\_enumext\_storing\_set:n and the function \\_\_enumext\_stop\_save\_ans\_msg: will be passed to the function \\_\_enumext\_execute\_after\_env:.

```

2076 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2077 {
2078     \msg_term:nnVV { enumext } { save-ans-log }
2079     \g__enumext_envir_name_tl \l__enumext_store_name_tl
2080 }
2081 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2082 {
2083     \msg_term:nnVV { enumext } { save-ans-log-hook }
2084     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2085 }

```

(End of definition for `\__enumext_start_save_ans_msg:` and `\__enumext_stop_save_ans_msg:`.)

`\__enumext_storing_set:n` The function `\__enumext_storing_set:n` first pass the value of the `save-ans` key to the variable `\l__enumext_store_name_tl` which will contain the `{⟨store name⟩}` of the *sequence* and *prop list* we will use. If `\l__enumext_store_name_tl` is *empty* we return an error message, otherwise will return the appropriate message `\__enumext_start_save_ans_msg:` and proceed to execute the function `\__enumext_storing_exec:` for `enumext` and `enumext*` environments.

```

2086 \cs_new_protected:Npn \__enumext_storing_set:n #1
2087 {
2088   \tl_set:Nx \l__enumext_store_name_tl {#1}
2089   \tl_if_empty:NTF \l__enumext_store_name_tl
2090   {
2091     \bool_lazy_or:nnT
2092       { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2093     {
2094       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2095     }
2096   }
2097   {
2098     \bool_lazy_or:nnT
2099       { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2100     {
2101       \__enumext_start_save_ans_msg:
2102       \__enumext_storing_exec:
2103     }
2104   }
2105 }

```

The function `\__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `anskey*`, `keyans`, `keyans*` and `keyanspic` environments and will set to “true” the variable `\l__enumext_check_answers_bool` used for internal checking answers mechanism set by the `check-ans` and `no-store` keys, copy `{⟨store name⟩}` into the variable `\g__enumext_store_name_tl` and execute the function `\__enumext_anskey_env_make:V` creating the environment `anskey*` (§13.31).

```

2106 \cs_new_protected:Nn \__enumext_storing_exec:
2107 {
2108   \bool_set_true:N \l__enumext_store_active_bool
2109   \bool_set_true:N \l__enumext_check_answers_bool
2110   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2111   \__enumext_anskey_env_make:V \l__enumext_store_name_tl

```

The *prop list* `\g__enumext_series_⟨store name⟩_prop` and the *sequence* `\g__enumext_series_⟨store name⟩_seq` will be created globally to “store content” in case they do not exist together with the integer variable `\g__enumext_series_⟨store name⟩_int` used by the keys `resume` and `resume*`.

```

2112 \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2113 {
2114   \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2115   \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2116 }
2117 \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2118 {
2119   \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2120   \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2121 }
2122 \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2123 {
2124   \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2125   \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2126 }
2127 }

```

(End of definition for `\__enumext_storing_set:n` and `\__enumext_storing_exec:`.)

### 13.26.3 The check answer mechanism

The internal mechanism for “checking answers” follows this logic:

If the line begins with `\item` or `\item*` and does NOT *open a nested environment*, each `\item` or `\item*` must contain a *single* execution of the `\anskey` command, i.e. the counter of the executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.



If the line begins with `\item` or `\item*` and *opens a nested environment* each `\item` or `\item*` in the nested environment must have a *single* execution of the `\anskey` command and the counter associated to the sum of `\item` and `\item*` executions must decrementing by “one” to maintain equality.

In order for the mechanism for the check-answer to work (not counting `keyans`, `keyans*` and `keyanspic`) we need:

1. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear within the environment including the nested levels.
2. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_item_number_int` must match the integer variable `\g__enumext_item_anskey_int` associated to the execution of the command `\anskey`. We analyze the cases:

- a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
- b) If the list has *nested levels*, for each level of nesting we need to decrementing by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.

With `keyans`, `keyans*` and `keyanspic` it is enough to increase in one the integer of `\anskey`. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a “hook” function after closing the *first level* of the environment.

### 13.26.4 Setting check-ans and no-store keys

Now we define the keys `check-ans` and `no-store` for all levels of `enumext` and `enumext*` environments.

```

check-ans 2128 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 2129 {
2130   \keys_define:nn { enumext / #1 }
2131   {
2132     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2133     check-ans .initial:n = false,
2134     check-ans .value_required:n = true,
2135     no-store .code:n = {
2136       \bool_set_false:N \l__enumext_check_answers_bool
2137       \bool_set_false:N \l__enumext_check_ans_key_bool
2138     },
2139     no-store .value_forbidden:n = true,
2140   }
2141 }
2142 \clist_map_inline:nn
2143 {
2144   level-1, level-2, level-3, level-4, enumext*
2145 }
2146 { \__enumext_tmp:n {#1} }
```

(End of definition for `check-ans` and `no-store`.)

### 13.26.5 Set-up check answer mechanism

The function `\__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `\__enumext_check_ans_level:` only if “true”, i.e. the key `no-store` is not active.

```

2147 \cs_new_protected:Nn \__enumext_check_ans_active:
2148 {
2149   \tl_if_empty:NF \l__enumext_store_name_tl
2150   {
2151     \bool_if:NT \l__enumext_check_answers_bool
2152     {
2153       \__enumext_check_ans_level:
2154     }
2155   }
2156 }
```

The function `\__enumext_check_ans_level:` will decrement by “one” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment `enumext`, taking into account whether it is nested within `enumext*` or the opposite and set `\l__enumext_item_number_bool` to “false”.

```

2157 \cs_new_protected:Nn \__enumext_check_ans_level:
2158 {
```

```

2159 \int_case:nn { \l__enumext_level_int }
2160 {
2161   { 1 }{
2162     \bool_lazy_all:nT
2163     {
2164       { \bool_if_p:N \g__enumext_starred_bool }
2165       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2166     }
2167     {
2168       \int_gdecr:N \g__enumext_item_number_int
2169       \bool_set_false:N \l__enumext_item_number_bool
2170     }
2171   }
2172   { 2 }{
2173     \int_gdecr:N \g__enumext_item_number_int
2174     \bool_set_false:N \l__enumext_item_number_bool
2175   }
2176   { 3 }{
2177     \int_gdecr:N \g__enumext_item_number_int
2178     \bool_set_false:N \l__enumext_item_number_bool
2179   }
2180   { 4 }{
2181     \int_gdecr:N \g__enumext_item_number_int
2182     \bool_set_false:N \l__enumext_item_number_bool
2183   }
2184 }

```

We should only execute this if `enumext*` is nested in the “*first level*” of `enumext`, for the rest of the cases the value of `\g__enumext_item_number_int` is already decreased.

```

2185 \int_case:nn { \l__enumext_level_h_int }
2186 {
2187   { 1 }{
2188     \bool_lazy_all:nT
2189     {
2190       { \bool_if_p:N \g__enumext_standar_bool }
2191       { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2192     }
2193     {
2194       \int_gdecr:N \g__enumext_item_number_int
2195       \bool_set_false:N \l__enumext_item_number_bool
2196     }
2197   }
2198 }
2199 }

```

(End of definition for `\__enumext_check_ans_active:` and `\__enumext_check_ans_level:`.)

`\__enumext_check_ans_key_hook:`

The function `\__enumext_check_ans_key_hook:` will *export* the status of the local variable `\l__enumext_check_ans_key_bool` to the global variable `\g__enumext_check_ans_key_bool` only if the key `check-ans` is active.

```

2200 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2201 {
2202   \bool_lazy_and:nnT
2203   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2204   { \bool_if_p:N \g__enumext_standar_bool }
2205   {
2206     \bool_gset_true:N \g__enumext_check_ans_key_bool
2207   }
2208   \bool_lazy_and:nnT
2209   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2210   { \bool_if_p:N \g__enumext_starred_bool }
2211   {
2212     \bool_gset_true:N \g__enumext_check_ans_key_bool
2213   }
2214 }

```

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

`\__enumext_item_answer_diff:`

The function `\__enumext_item_answer_diff:` will set the value of the variable `\g__enumext_item_answer_diff_int` which is used by the functions `\__enumext_check_ans_show:` for the key `save-ans`

and by the function `\__enumext_check_ans_log`: by the internal “*check answer*” mechanism. This function will be passed to the function `\__enumext_execute_after_env`:

```

2215 \cs_new_protected:Nn \__enumext_item_answer_diff:
2216 {
2217   \int_gset:Nn \g__enumext_item_answer_diff_int
2218   {
2219     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2220   }
2221 }

```

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

```

\__enumext_check_ans_show:
  \__enumext_check_ans_msg_less:
  \__enumext_check_ans_msg_same_ok:
  \__enumext_check_ans_msg_greater:

```

The function `\__enumext_check_ans_show`: will be executed within the function `\__enumext_execute_after_env`: when the key `check-ans` is active, that is, when `\g__enumext_check_ans_key_bool` is “*true*” and will return the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `\__enumext_item_answer_diff`:

```

2222 \cs_new_protected:Nn \__enumext_check_ans_show:
2223 {
2224   \int_case:nn { \g__enumext_item_answer_diff_int }
2225   {
2226     { -1 } { \__enumext_check_ans_msg_less: }
2227     { 0 } { \__enumext_check_ans_msg_same_ok: }
2228     { 1 } { \__enumext_check_ans_msg_greater: }
2229   }
2230 }
2231 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2232 {
2233   \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2234   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2235 }
2236 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2237 {
2238   \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2239   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2240 }
2241 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2242 {
2243   \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2244   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2245 }

```

(End of definition for `\__enumext_check_ans_show`: and others.)

```

\__enumext_check_ans_log:
  \__enumext_check_ans_log_msg_less:
  \__enumext_check_ans_log_msg_same_ok:
  \__enumext_check_ans_log_msg_greater:

```

The function `\__enumext_check_ans_log`: will be executed within the function `\__enumext_execute_after_env`: when the key `check-ans` is not active, that is, when `\g__enumext_check_ans_key_bool` is “*false*” and write in the log the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `\__enumext_item_answer_diff`:

```

2246 \cs_new_protected:Nn \__enumext_check_ans_log:
2247 {
2248   \int_case:nn { \g__enumext_item_answer_diff_int }
2249   {
2250     { -1 } { \__enumext_check_ans_log_msg_less: }
2251     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2252     { 1 } { \__enumext_check_ans_log_msg_greater: }
2253   }
2254 }
2255 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2256 {
2257   \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2258   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2259 }
2260 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2261 {
2262   \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2263   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2264 }
2265 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2266 {
2267   \msg_log:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2268   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2269 }

```

(End of definition for `\__enumext_check_ans_log:` and others.)

### 13.26.6 Check for `\item*` and `\anspic*` commands

`\__enumext_check_starred_cmd:n`

The function `\__enumext_check_starred_cmd:n` performs an *extra check* for the `keyans`, `keyans*` and `keyanspic` environments. Unlike the *check* executed by `check-ans` key this one is not controlled by any key, it is intended to prevent the forgetting of `\item*` or `\anspic*` in these environments.

```

2270 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2271 {
2272   \int_compare:nNtT
2273     { \g__enumext_check_starred_cmd_int } = { 0 }
2274   {
2275     \msg_warning:nnnV
2276       { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2277   }
2278   \int_compare:nNtT
2279     { \g__enumext_check_starred_cmd_int } > { 1 }
2280   {
2281     \msg_warning:nnnV
2282       { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2283   }
2284   \int_gzero:N \g__enumext_check_starred_cmd_int
2285   \tl_clear:N \l__enumext_check_start_line_env_tl
2286 }

```

(End of definition for `\__enumext_check_starred_cmd:n`.)

### 13.27 Keys and functions associated with storage

wrap-ans  
wrap-opt  
save-sep  
mark-ans  
mark-pos  
show-ans  
mark-ref  
save-ref

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “storage system” and internal mechanism of “label and ref” only at the *first level* of `enumext` and `enumext*`.

```

2287 \cs_set_protected:Npn \__enumext_tmp:n #1
2288 {
2289   \keys_define:nn { enumext / #1 }
2290   {
2291     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,
2292     wrap-ans .initial:n =
2293       {
2294         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2295       },
2296     wrap-ans .value_required:n = true,
2297     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n #1,
2298     wrap-opt .initial:n = [{##1}],
2299     wrap-opt .value_required:n = true,
2300     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2301     save-sep .initial:n = { , ~ },
2302     save-sep .value_required:n = true,
2303     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2304     mark-ans .initial:n = \textasteriskcentered,
2305     mark-ans .value_required:n = true,
2306     mark-pos .choice:,
2307     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2308     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2309     mark-pos / unknown .code:n =
2310       \msg_error:nnee { enumext } { unknown-choice }
2311       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2312     mark-pos .initial:n = right,
2313     mark-pos .value_required:n = true,
2314     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2315     show-ans .initial:n = false,
2316     show-ans .value_required:n = true,
2317     show-pos .bool_set:N = \l__enumext_show_position_bool,
2318     show-pos .initial:n = false,
2319     show-pos .value_required:n = true,
2320     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2321     mark-ref .initial:n = \textreferencemark,
2322     mark-ref .value_required:n = true,
2323     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2324     save-ref .initial:n = false,
2325     save-ref .value_required:n = true,
2326   }

```

```

2327 }
2328 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for wrap-ans and others.)

For the `keyans` and `keyans*` environments we will only add the keys `mark-pos`, `show-ans` and `show-pos`.

```

mark-pos \cs_set_protected:Npn \__enumext_tmp:n #1
show-ans {
show-pos {
2329 \keys_define:nn { enumext / #1 }
2330 {
2331 {
2332 mark-pos .choice:,
2333 mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2334 mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2335 mark-pos .initial:n = right,
2336 mark-pos .value_required:n = true,
2337 show-ans .bool_set:N = \l__enumext_show_answer_bool,
2338 show-ans .initial:n = false,
2339 show-ans .value_required:n = true,
2340 show-pos .bool_set:N = \l__enumext_show_position_bool,
2341 show-pos .initial:n = false,
2342 show-pos .value_required:n = true,
2343 }
2344 }
2345 }
2346 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

(End of definition for mark-pos, show-ans, and show-pos.)

### 13.27.1 Storing structure of the environments

The idea behind “*storing structure*” in the *sequence* is to have a copy of the *structure of the environment* in which the key `save-ans` is being executed so we must capture the *optional argument* passed to the levels of the environment in which it is executed and “*storing*” this in the *sequence*.

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

The functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n` will be responsible for the “*storing keys*” filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `\l__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered *<keys>*.

```

2347 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2348 {
2349 \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2350 {
2351 \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2352 \tl_set:ce
2353 { \l__enumext_store_save_key_ \__enumext_level: _tl }
2354 { \__enumext_filter_save_key:n {#1} }
2355 }
2356 }
2357 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2358 {
2359 \bool_if:NF \l__enumext_store_save_key_vii_bool
2360 {
2361 \tl_clear:N \l__enumext_store_save_key_vii_tl
2362 \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2363 }
2364 }

```

(End of definition for `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`.)

### 13.27.2 Setting save-key key

Since this “*storing structure*” in the *sequence* established by the `save-ans` key when executing `\anskey` or `anskey*`, we will not be able to modify it. The best thing here is to have a key that allows you to modify the *optional argument* of the “*storing structure*” in the *sequence*.

The values set by this key passed in the *optional argument* of the `enumext` and `enumext*` environments will override the values of the `\l__enumext_store_save_key_X_tl` variable set by the functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`. Now define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2365 \cs_set_protected:Npn \__enumext_tmp:n #1
2366 {

```

```

2367 \keys_define:nn { enumext / enumext* }
2368 {
2369   save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2370   save-key .value_required:n = true,
2371 }
2372 \keys_define:nn { enumext / #1 }
2373 {
2374   save-key .code:n = \__enumext_parse_save_key:n {##1},
2375   save-key .value_required:n = true,
2376 }
2377 }
2378 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

```

\__enumext_parse_save_key:n
\__enumext_parse_save_key_vii:n

```

The functions `\__enumext_parse_save_key:n` and `\__enumext_parse_save_key_vii:n` will be responsible for “storing keys” in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2379 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2380 {
2381   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2382   \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2383   \tl_set:ce
2384     { l__enumext_store_save_key_ \__enumext_level: _tl }
2385     { \__enumext_filter_save_key:n {#1} }
2386 }
2387 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2388 {
2389   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2390   \tl_clear:N \l__enumext_store_save_key_vii_tl
2391   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2392 }

```

(End of definition for `\__enumext_parse_save_key:n` and `\__enumext_parse_save_key_vii:n`.)

### 13.27.3 Internal functions to store optional arguments

```

\__enumext_filter_save_key:n
\__enumext_filter_save_key_key:n
\__enumext_filter_save_key_pair:nn

```

The function `\__enumext_filter_save_key:n` will be in charge of “filtering keys” we want to stored in sequence where `{#1}` represents the *optional argument* passed to the environment.

```

2393 \cs_new:Npn \__enumext_filter_save_key:n #1
2394 {
2395   \use:e
2396   {
2397     \keyval_parse:NNn
2398       \__enumext_filter_save_key_key:n
2399       \__enumext_filter_save_key_pair:nn {#1}
2400   }
2401 }

```

The function `\__enumext_filter_save_key_key:n` will be responsible for “filtering keys” that are passed “without value” by excluding the `resume`, `resume*`, `no-store` and `base-fix` keys.

```

2402 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2403 {
2404   \str_case:nnF {#1}
2405   {
2406     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2407   }
2408   { , { \exp_not:n {#1} } }
2409 }

```

The function `\__enumext_filter_save_key_pair:nn` will be responsible for “filtering keys” that are passed “with value” by excluding the `series`, `resume`, `save-ans`, `save-ref`, `check-ans`, `show-ans`, `save-pos`, `wrap-ans`, `mark-ans`, `wrap-opt`, `save-sep`, `mark-ref`, `mini-env`, `mini-sep`, `mini-right` and `mini-right*` keys.

```

2410 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2411 {
2412   \str_case:nnF {#1}
2413   {
2414     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2415     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2416     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2417     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2418     { mini-right* } {}

```

```

2419     }
2420     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2421 }

```

(End of definition for `\__enumext_filter_save_key:n`, `\__enumext_filter_save_key_key:n`, and `\__enumext_filter_save_key_pair:nn`.)

### 13.27.4 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

The function `\__enumext_store_addto_prop:n` stores the  $\{\langle content \rangle\}$  in *prop list* defined by `save-ans` key. The “stored content” is retrieved by means of the `\getkeyans` command.

The form in which the  $\{\langle content \rangle\}$  is “stored” in the *prop list* is  $\{\langle position \rangle\}\{\langle content \rangle\}$ . This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

2422 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2423 {
2424   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2425   {
2426     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2427   }
2428   { #1 }
2429 }
2430 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

(End of definition for `\__enumext_store_addto_prop:n`.)

### 13.27.5 Function for storing content in sequence

```

\__enumext_store_addto_seq:n
\__enumext_store_addto_seq:v
\__enumext_store_addto_seq:V

```

The function `\__enumext_store_addto_seq:n` stores the  $\{\langle content \rangle\}$  in *sequence* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

The form in which the  $\{\langle content \rangle\}$  is stored in *sequence* is in a internal `enumext` or `enumext*` environments with the “same structure” in which the command was executed.

The “stored content” is retrieved by means of the `\printkeyans` command.

```

2431 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2432 {
2433   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2434 }
2435 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

(End of definition for `\__enumext_store_addto_seq:n`.)

### 13.27.6 Functions for storing structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

The “storing structure” is handled by the functions `\__enumext_store_level_open:` and `\__enumext_store_level_close:` which are executed per level within the `enumext` environment.

```

2436 \cs_new_protected:Nn \__enumext_store_level_open:
2437 {
2438   \bool_if:NT \l__enumext_check_answers_bool
2439   {
2440     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2441     {
2442       \__enumext_store_addto_seq:n
2443       {
2444         \item \begin{enumext}
2445       }
2446     }
2447     {
2448       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2449       {
2450         \item \begin{enumext} [
2451         ]
2452         \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2453         {
2454           ]
2455         }
2456         \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2457       }
2458     }
2459   }
2460 \cs_new_protected:Nn \__enumext_store_level_close:
2461 {
2462   \bool_if:NT \l__enumext_check_answers_bool
2463   {

```



```

2464         \__enumext_store_addto_seq:n { \end{enumext} }
2465     }
2466 }

```

(End of definition for \\_\_enumext\_store\_level\_open: and \\_\_enumext\_store\_level\_close:.)

```

\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:

```

The “*storing structure*” is handled by the functions \\_\_enumext\_store\_level\_open\_vii: and \\_\_enumext\_store\_level\_close\_vii: which are executed in the `enumext*` environment.

```

2467 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2468 {
2469     \bool_if:NT \__enumext_check_answers_bool
2470     {
2471         \tl_if_empty:NTF \__enumext_store_save_key_vii_tl
2472         {
2473             \__enumext_store_addto_seq:n
2474             {
2475                 \item \begin{enumext*}
2476             }
2477         }
2478         {
2479             \tl_put_left:Nn \__enumext_store_save_key_vii_tl
2480             {
2481                 \item \begin{enumext*}[
2482             }
2483             \tl_put_right:Nn \__enumext_store_save_key_vii_tl
2484             {
2485                 ]
2486             }
2487             \__enumext_store_addto_seq:V \__enumext_store_save_key_vii_tl
2488         }
2489     }
2490 }
2491 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2492 {
2493     \bool_if:NT \__enumext_check_answers_bool
2494     {
2495         \__enumext_store_addto_seq:n { \end{enumext*} }
2496     }
2497 }

```

(End of definition for \\_\_enumext\_store\_level\_open\_vii: and \\_\_enumext\_store\_level\_close\_vii:.)

### 13.27.7 Function for show marks and position

```

\__enumext_print_keyans_box:NN
\__enumext_print_keyans_box:cc

```

The function \\_\_enumext\_print\_keyans\_box:NN print a box in the left margin with \l\_\_enumext\_mark\_answer\_sym\_tl used by the `wrap-ans`, `show-ans` and `show-pos` keys. The function takes two arguments:

```

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```

```

2498 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2499 {
2500     \mode_leave_vertical:
2501     \skip_horizontal:n { -\dim_use:N #2 }
2502     \makebox[0pt][ r ]
2503     {
2504         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2505         {
2506             \tl_use:N \l__enumext_mark_answer_sym_tl
2507         }
2508     }
2509     \skip_horizontal:n { \dim_use:N #2 }
2510 }
2511 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for \\_\_enumext\_print\_keyans\_box:NN.)

### 13.28 The internal label and ref

The function \\_\_enumext\_store\_internal\_ref: handles the “*internal label and ref*” system used by the `save-ref` and `mark-ref` keys for \anskey will allow to execute `\ref{⟨store name : position⟩}` and will return `1.(a).i.A`.

\\\_enumext\_store\_internal\_ref:

First we will remove the dots “.” from the current  $\langle labels \rangle$ , we do not want to get double dots in our references, then we will place this in the variable \\\_enumext\_newlabel\_arg\_two\_tl.

```

2512 \cs_new_protected:Nn \\_enumext_store_internal_ref:
2513 {
2514   \cs_set_protected:Npn \\_enumext_tmp:n ##1
2515   {
2516     \tl_set_eq:cc { \\_enumext_label_copy_##1_tl } { \\_enumext_label_##1_tl }
2517     \tl_reverse:c { \\_enumext_label_copy_##1_tl }
2518     \tl_remove_once:cn { \\_enumext_label_copy_##1_tl } { . }
2519     \tl_reverse:c { \\_enumext_label_copy_##1_tl }
2520   }
2521   \clist_map_inline:nn { i, ii, iii, iv, vii } { \\_enumext_tmp:n {##1} }
2522   \cs_set:Npn \\_enumext_tmp:n ##1
2523   { . \tl_use:c { \\_enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

Here we need to analyse the cases where the environment is started with `enumext*` and if `\anskey` or `anskey*` is running alone in it or if it is running in a nested `enumext` environment within the starting environment.

```

2524   \bool_lazy_all:nT
2525   {
2526     { \bool_if_p:N \g__enumext_starred_bool }
2527     { \int_compare_p:nNn { \\_enumext_level_int } = { 0 } }
2528   }
2529   {
2530     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2531     { \tl_use:N \\_enumext_label_copy_vii_tl }
2532   }
2533   \bool_lazy_all:nT
2534   {
2535     { \bool_not_p:n { \g__enumext_standar_bool } }
2536     { \bool_if_p:N \\_enumext_standar_bool }
2537     { \int_compare_p:nNn { \\_enumext_level_int } > { 0 } }
2538   }
2539   {
2540     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2541     {
2542       \tl_use:N \\_enumext_label_copy_vii_tl
2543       \int_step_function:nnN { 1 } { \\_enumext_level_int } \\_enumext_tmp:n
2544     }
2545   }

```

If started with `enumext` and if `\anskey` or `anskey*` is running alone in it or if it is running in a nested `enumext*` environment within the starting environment.

```

2546   \bool_lazy_all:nT
2547   {
2548     { \bool_if_p:N \g__enumext_standar_bool }
2549     { \int_compare_p:nNn { \\_enumext_level_int } > { 0 } }
2550     { \int_compare_p:nNn { \\_enumext_level_h_int } = { 0 } }
2551   }
2552   {
2553     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2554     {
2555       \tl_use:N \\_enumext_label_copy_i_tl
2556       \int_step_function:nnN { 2 } { \\_enumext_level_int } \\_enumext_tmp:n
2557     }
2558   }
2559   \cs_set:Npn \\_enumext_tmp:n ##1
2560   { \tl_use:c { \\_enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2561   \bool_lazy_all:nT
2562   {
2563     { \bool_if_p:N \g__enumext_standar_bool }
2564     { \bool_if_p:N \\_enumext_starred_bool }
2565     { \int_compare_p:nNn { \\_enumext_level_int } > { 0 } }
2566   }
2567   {
2568     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2569     {
2570       \int_step_function:nnN { 1 } { \\_enumext_level_int } \\_enumext_tmp:n
2571       \tl_use:N \\_enumext_label_copy_vii_tl
2572     }
2573   }

```

Now we set the variable \\\_enumext\_newlabel\_arg\_one\_tl which will contain  $\{ \langle store name : position \rangle \}$ .

```

2574 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2575 {
2576   \l__enumext_store_name_tl \c_colon_str
2577   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2578 }

```

Now execute the function `\__enumext_newlabel:nn` and save the result in the variable `\l__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```

2579 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2580 {
2581   \__enumext_newlabel:nn
2582   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2583   { \l__enumext_newlabel_arg_two_tl }
2584 }
2585 \l__enumext_write_aux_file_tl
2586 }

```

(End of definition for `\__enumext_store_internal_ref:`)

### 13.29 Common functions for `\anskey` and `anskey*` environment

`\__enumext_store_anskey_code:n`

The internal function `\__enumext_store_anskey_code:n` first we pass the  $\langle \textit{argument} \rangle$  to the *prop list*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the *save-ref* key and will call the function `\__enumext_store_internal_ref:` for the “*internal label and ref*” system. Followed by this if the *show-ans* or *show-pos* keys are active we will show the “*wrapped*”  $\langle \textit{argument} \rangle$ .

```

2587 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2588 {
2589   \int_gincr:N \g__enumext_item_anskey_int
2590   \__enumext_store_addto_prop:n {#1}
2591   \bool_if:NT \l__enumext_store_ref_key_bool
2592   {
2593     \__enumext_store_internal_ref:
2594   }
2595   \__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the  $\llbracket \textit{key} = \textit{val} \rrbracket$  passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “*store*” in the *sequence*. First we clear the variable `\l__enumext_store_anskey_arg_tl` and process the  $\langle \textit{keys} \rangle$ , if the *break-col* key is present and the command is running under *enumext* (not in *enumext\**) we will add `\columnbreak` and then `\item`.

```

2596 \tl_clear:N \l__enumext_store_anskey_arg_tl
2597 \bool_lazy_and:nnT
2598 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2599 { \bool_not_p:n { \l__enumext_starred_bool } }
2600 {
2601   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2602 }
2603 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

If the *item-join* key is present and the command is running under *enumext\** we will add  $\langle \textit{number} \rangle$  to `\l__enumext_store_anskey_arg_tl`.

```

2604 \bool_lazy_and:nnT
2605 { \bool_not_p:n { \l__enumext_starred_bool } }
2606 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2607 {
2608   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2609   {
2610     ( \exp_not:V \l__enumext_store_item_join_int )
2611   }
2612 }

```

And now we will review the keys *item-star*, *item-sym\** and *item-pos\** and pass them to `\l__enumext_store_anskey_arg_tl` along with the  $\langle \textit{argument} \rangle$  for `\anskey` or  $\langle \textit{body} \rangle$  for *anskey\**.

```

2613 \bool_if:NTF \l__enumext_store_item_star_bool
2614 {
2615   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2616   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2617   {
2618     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2619     {
2620       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2621     }
2622   }
2623   \dim_compare:nT

```

```

2624         {
2625             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2626         }
2627         {
2628             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2629             {
2630                 [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2631             }
2632         }
2633         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2634     }
2635     {
2636         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2637     }

```

Finally we check if the `save-ref` key are active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` with “*symbol*” set by `mark-ref` key and then store in *sequence*.

```

2638     \bool_lazy_and:nnT
2639     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2640     { \bool_if_p:N \l__enumext_hyperref_bool }
2641     {
2642         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2643         {
2644             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2645             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2646         }
2647     }
2648     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2649 }

```

(End of definition for `\__enumext_store_anskey_code:n`.)

`\__enumext_anskey_show_wrap_arg:n`

The function `\__enumext_anskey_show_wrap_arg:n` “wraps” the  $\langle argument \rangle$  passed to `\anskey` and the  $\langle body \rangle$  for `anskey*` when using the `wrap-ans` key.

```

2650 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2651 {
2652     \par
2653     \bool_if:NTF \l__enumext_starred_bool
2654     {
2655         \__enumext_print_keyans_box:NN
2656         \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2657     }
2658     {
2659         \__enumext_print_keyans_box:cc
2660         { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2661         { \l__enumext_labelsep_ \l__enumext_level: _dim }
2662     }
2663     \__enumext_anskey_wrapper:n { #1 }
2664 }

```

(End of definition for `\__enumext_anskey_show_wrap_arg:n`.)

`\__enumext_anskey_show_wrap_left:n`

The function `\__enumext_anskey_show_wrap_left:n` will show the “*mark*” defined by the `mark-ans` key or the “*position*” of the  $\langle content \rangle$  stored in the *prop list* when using the `show-pos` key on the left margin next to the “wraps”  $\langle argument \rangle$  passed to `\anskey` and the  $\langle body \rangle$  in `anskey*` on the right side when using the `show-ans` key.

```

2665 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2666 {
2667     \bool_if:NT \l__enumext_show_answer_bool
2668     {
2669         \__enumext_anskey_show_wrap_arg:n { #1 }
2670     }
2671     \bool_if:NT \l__enumext_show_position_bool
2672     {
2673         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2674         {
2675             \group_begin:
2676             \exp_not:N \normalfont
2677             \exp_not:N \footnotesize [ \int_eval:n
2678                 {
2679                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }

```

```

2680         }
2681     ]
2682     \group_end:
2683 }
2684 \__enumext_anskey_show_wrap_arg:n { #1 }
2685 }
2686 }

```

(End of definition for \\_\_enumext\_anskey\_show\_wrap\_left:n.)

### 13.30 The command \anskey

Since we will be “*storing content*” in a `list` environment within *sequences* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing.

The `\anskey` command will cover this point and give it similar behaviour to that of `\item` in the `enumext` and `enumext*` environments executed as follows `\anskey[⟨key = val⟩]{⟨content⟩}`.

First we’ll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```

2687 \keys_define:nn { enumext / anskey }
2688 {
2689     break-col .bool_set:N = \__enumext_store_columns_break_bool,
2690     break-col .default:n = true,
2691     break-col .value_forbidden:n = true,
2692     item-join .int_set:N = \__enumext_store_item_join_int,
2693     item-join .value_required:n = true,
2694     item-star .bool_set:N = \__enumext_store_item_star_bool,
2695     item-star .default:n = true,
2696     item-star .value_forbidden:n = true,
2697     item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
2698     item-sym* .value_required:n = true,
2699     item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
2700     item-pos* .value_required:n = true,
2701     unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2702 }

```

The `⟨keys⟩` are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `\__enumext_anskey_unknown:n`.

```

2703 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2704 {
2705     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2706 }
2707 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2708 {
2709     \tl_if_blank:nTF {#2}
2710     {
2711         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2712     }
2713     {
2714         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2715     }
2716 }

```

(End of definition for \\_\_enumext\_anskey\_unknown:n and \\_\_enumext\_anskey\_unknown:nn.)

- The `\anskey` command will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error.

`\anskey` We will first call the function `\__enumext_anskey_safe_outer:` to be sure where we execute the command, then we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, if is true we will increment `\g__enumext_item_anskey_int` for the internal “*check answer*” system and execute the function `\__enumext_anskey_safe_inner:n` to ensure that the command is not nested and that the argument is not empty, finally search the `[⟨key = val⟩]` and call the function `\__enumext_store_anskey_code:n`.

```

2717 \NewDocumentCommand \anskey { o +m }
2718 {
2719     \__enumext_anskey_safe_outer:
2720     \group_begin:
2721         \bool_if:NT \l__enumext_check_answers_bool
2722         {
2723             \tl_if_novalue:nF {#1}
2724             {
2725                 \keys_set:nn { enumext / anskey } {#1}

```

```

2726     }
2727     \tl_if_blank:nTF {#2}
2728     {
2729         \msg_error:nn { enumext } { anskey-empty-arg }
2730     }
2731     {
2732         \__enumext_anskey_safe_inner:
2733         \__enumext_store_anskey_code:n {#2}
2734     }
2735 }
2736 \group_end:
2737 }

```

(End of definition for `\anskey`. This function is documented on page 13.)

### 13.30.1 Internal functions for the command

```

\__enumext_anskey_safe_outer:
\__enumext_anskey_safe_inner:

```

The `\__enumext_store_anskey_safe_outer:` function will return the appropriate messages when the command is executed outside the environment in which the `save-ans` key was activated.

```

2738 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2739 {
2740     \bool_if:NF \l__enumext_store_active_bool
2741     {
2742         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2743     }
2744     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2745     {
2746         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2747     }
2748     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2749     {
2750         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2751     }
2752     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2753     {
2754         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2755     }
2756 }

```

The `\__enumext_anskey_safe_inner:` function will first check if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2757 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2758 {
2759     \int_incr:N \l__enumext_anskey_level_int
2760     \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2761     {
2762         \msg_error:nn { enumext } { anskey-nested }
2763     }
2764     \bool_if:NF \l__enumext_item_number_bool
2765     {
2766         \msg_error:nn { enumext } { anskey-unnumber-item }
2767     }
2768     \mode_if_math:T
2769     {
2770         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2771     }
2772 }

```

(End of definition for `\__enumext_anskey_safe_outer:` and `\__enumext_anskey_safe_inner:`.)

### 13.31 The environment `anskey*`

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the `scontents` package, so to be able to have support at this point it is best to play a little with the internal code of `scontents` and *hooks*. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to `store-env`, if you are using the key `print-env` or the `write-out` key, sure, I can detect and modify it within the `enumext` and `enumext*` environments, but it would look strange not to have some keys available when running within these environments.

- A better (perhaps a bit paranoid) option is to define it within the environment in which the `save-ans` key is executed. and have it available only when that key is executed, here I would have absolute control of the `(keys)` and I make sure that `write-out` is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

`\__enumext_undefine_anskey_env:`

The function `\__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `\__enumext_execute_after_env:` (§13.32) which is executed after the environment in which the key `save-ans` is active.

```
2773 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2774 {
2775   \cs_undefine:c { anskey* }
2776   \cs_undefine:c { endanskey* }
2777   \cs_undefine:c { __scontents_anskey*_env_begin: }
2778   \cs_undefine:c { __scontents_anskey*_env_end: }
2779 }
```

Detection of the `anskey*` environment outside the `enumext` and `enumext*` environments.

```
2780 \__enumext_before_env:nn { enumext }
2781 {
2782   \bool_lazy_and:nnT
2783   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2784   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2785   {
2786     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2787     {
2788       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2789     }
2790   }
2791 }
2792 \__enumext_before_env:nn { enumext* }
2793 {
2794   \bool_lazy_and:nnT
2795   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2796   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2797   {
2798     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2799     {
2800       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2801     }
2802   }
2803 }
```

Detection of the `anskey*` environment inside the `keyans`, `keyans*` and `keyanspic` environments, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```
2804 \__enumext_before_env:nn { anskey* }
2805 {
2806   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2807   {
2808     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2809   }
2810   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2811   {
2812     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2813   }
2814   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2815   {
2816     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2817   }
2818   \bool_if:NF \__enumext_item_number_bool
2819   {
2820     \msg_error:nn { enumext } { anskey-unnumber-item }
2821   }
2822   \mode_if_math:T
2823   {
2824     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2825   }
2826 }
```

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



**anskey\***

The function `\__enumext_anskey_env_make:n` creates the environment **anskey\*** (custom version of **scontents** environment) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`.

To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `\__enumext_storing_exec:` (§13.26.1) and we will execute it only if the variable `\l__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package **scontents**.

```

2827 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2828 {
2829   \bool_if:NT \l__enumext_anskey_env_bool
2830   {
2831     \newenvsc{anskey*}[store-env=#1,print-env=false]
2832     \__enumext_anskey_env_exec:
2833   }
2834 }
2835 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }

```

The function `\__enumext_anskey_env_define_keys:` will add the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` and will leave the keys `print-env`, `store-env` and `write-out` undefined. We will apply this function using the *hook* function `\__enumext_before_env:nn`.

```

2836 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2837 {
2838   \keys_define:nn { scontents / scontents }
2839   {
2840     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2841     break-col .default:n   = true,
2842     break-col .value_forbidden:n = true,
2843     item-join .int_gset:N   = \g__enumext_store_item_join_int,
2844     item-join .value_required:n = true,
2845     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2846     item-star .default:n   = true,
2847     item-star .value_forbidden:n = true,
2848     item-sym* .tl_gset:N    = \g__enumext_store_item_symbol_tl,
2849     item-sym* .value_required:n = true,
2850     item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2851     item-pos* .value_required:n = true,
2852     print-env .undefine:,
2853     store-env .undefine:,
2854     write-out .undefine:,
2855     unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2856   }
2857 }

```

The *⟨keys⟩* are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `\__enumext_anskey_env_unknown:n`.

```

2858 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2859 {
2860   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2861 }
2862 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2863 {
2864   \tl_if_blank:nTF {#2}
2865   {
2866     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2867   }
2868   {
2869     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2870   }
2871 }

```

The function `\__enumext_anskey_env_reset_keys:` will leave the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` undefined. We will apply this function using the *hook* function `\__enumext_after_env:nn`.

```

2872 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2873 {
2874   \keys_define:nn { scontents / scontents }
2875   {
2876     break-col .undefine:,
2877     item-join .undefine:,
2878     item-star .undefine:,
2879     item-sym* .undefine:,

```

```

2880     item-pos* .undefine:,
2881     write-out .code:n = {
2882         \bool_set_false:N \l__scontents_storing_bool
2883         \bool_set_true:N \l__scontents_writing_bool
2884         \tl_set:Nn \l__scontents_fname_out_tl {##1}
2885     },
2886     write-out .value_required:n = true,
2887     print-env .meta:nn = { scontents } { print-env = ##1 },
2888     print-env .default:n = true,
2889     store-env .meta:nn = { scontents } { store-env = ##1 },
2890     unknown .code:n = { \l__scontents_parse_environment_keys:n {##1} },
2891 }
2892 }

```

The function `\__enumext_rescan_anskey_env:n` will be responsible for bringing the *(body)* of the environment saved in the sequence `\g__scontents_name_{store name}_seq` to pass it to our *sequence* and *prop list*.

```

2893 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2894 {
2895     \group_begin:
2896     \int_set:Nn \tex_newlinechar:D { `^^J }
2897     \l__scontents_rescan_tokens:x
2898     {
2899         \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2900         #1
2901     }
2902 }

```

(End of definition for *anskey\** and others. This function is documented on page 14.)

`\__enumext_anskey_env_exec:`

The function `\__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our *(keys)*.

```

2903 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2904 {
2905     \__enumext_before_env:nn { anskey* }
2906     {
2907         \__enumext_anskey_env_define_keys:
2908     }

```

Now we will execute our actions after the *anskey\** environment is closed. We'll fetch the contents of the *environment body* that is now saved in `\g__scontents_name_{store name}_seq` and store it in the variable `\l__enumext_store_anskey_env_tl` then we execute the rest of the functions.

```

2909     \hook_if_empty:nF {env/anskey*/after}
2910     {
2911         \hook_gremove_code:nn {env/anskey*/after} { * }
2912     }
2913     \__enumext_after_env:nn { anskey* }
2914     {
2915         \__enumext_anskey_env_save_keys:
2916         \tl_clear:N \l__enumext_store_anskey_env_tl
2917         \tl_clear:N \l__enumext_store_anskey_opt_tl
2918         \bool_if:NT \l__enumext_check_answers_bool
2919         {
2920             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2921             {
2922                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2923             }
2924             \regex_match:nVTF
2925             { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2926             \l__enumext_store_anskey_env_tl
2927             {
2928                 \msg_error:nn { enumext } { anskey-empty-arg }
2929             }
2930             {
2931                 \__enumext_anskey_env_store:
2932             }
2933         }
2934         \__enumext_anskey_env_clean_vars:
2935         \__enumext_anskey_env_reset_keys:
2936     }
2937 }

```

- The use of `\hook_gremove_code:nn` is necessary here, otherwise the `{\code}` passed to `\__enumext_after_env:nn{anskey*}` will be accumulated for each execution. The last function `\__enumext_anskey_env_reset_keys:` is necessary so as not to hinder any `scontents` environment running within `enumext` or `enumext*`.

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

```
\__enumext_anskey_env_save_keys:
\__enumext_anskey_env_store:
\__enumext_anskey_env_clean_vars:
```

The function `\__enumext_anskey_env_save_keys:` processing the `[key = val]` passed to the environment and save this in the variable `\__enumext_store_anskey_opt_tl`. If the `break-col` key is present and the environment is running under `enumext` (not in `enumext*`) we will add the key `break-col`.

```
2938 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2939 {
2940   \bool_lazy_and:nnT
2941   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2942   { \bool_not_p:n { \l__enumext_starred_bool } }
2943   {
2944     \tl_put_left:Ne \__enumext_store_anskey_opt_tl { ,break-col, }
2945   }
2946 }
```

If the `item-join` key is present and the command is running under `enumext*` we will add to `\__enumext_store_anskey_opt_tl`.

```
2946 \bool_lazy_and:nnT
2947 { \bool_not_p:n { \l__enumext_starred_bool } }
2948 { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2949 {
2950   \tl_put_left::Ne \__enumext_store_anskey_opt_tl
2951   {
2952     ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2953   }
2954 }
```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\__enumext_store_anskey_opt_tl`.

```
2955 \bool_if:NT \g__enumext_store_item_star_bool
2956 {
2957   \tl_put_left:Ne \__enumext_store_anskey_opt_tl
2958   {
2959     ,item-star,
2960   }
2961   \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2962   {
2963     \tl_put_left:Ne \__enumext_store_anskey_opt_tl
2964     {
2965       ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2966     }
2967   }
2968   \dim_compare:nT
2969   {
2970     \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2971   }
2972   {
2973     \tl_put_left:Ne \__enumext_store_anskey_opt_tl
2974     {
2975       ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2976     }
2977   }
2978 }
2979 }
```

The function `\__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `\__enumext_store_anskey_code:n` and `\__enumext_rescan_anskey_env:n`.

```
2980 \cs_new_protected:Nn \__enumext_anskey_env_store:
2981 {
2982   \group_begin:
2983   \tl_if_empty:NTF \__enumext_store_anskey_opt_tl
2984   {
2985     \exp_args:Ne
2986     \__enumext_store_anskey_code:n
2987     {
2988       \__enumext_rescan_anskey_env:n { \__enumext_store_anskey_env_tl }
2989     }
2990   }
2991   {
```

```

2992         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2993         \exp_args:Ne
2994         \__enumext_store_anskey_code:n
2995         {
2996             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2997         }
2998     }
2999     \group_end:
3000 }

```

The function `\__enumext_anskey_env_clean_vars:` will return the global variables used by the `(keys)` to their initial state.

```

3001 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
3002 {
3003     \bool_gset_false:N \g__enumext_store_columns_break_bool
3004     \int_gzero:N       \g__enumext_store_item_join_int
3005     \bool_gset_false:N \g__enumext_store_item_star_bool
3006     \tl_gclear:N       \g__enumext_store_item_symbol_tl
3007     \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
3008 }

```

(End of definition for `\__enumext_anskey_env_save_keys:`, `\__enumext_anskey_env_store:`, and `\__enumext_anskey_env_clean_vars:`.)

### 13.32 Executing anskey\*, check-ans and write .log

`\__enumext_execute_after_env:`

The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `\__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `\__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `\__enumext_check_ans_log:` and write the results in the `.log` file, undefine the environment `anskey*` (§13.31) through the function `\__enumext_undefine_anskey_env:` and finally we execute the function `\__enumext_reset_global_vars:` returning the used variables to their original state.

```

3009 \cs_new_protected:Nn \__enumext_execute_after_env:
3010 {
3011     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3012     {
3013         \tl_if_empty:NF \g__enumext_store_name_tl
3014         {
3015             \__enumext_stop_save_ans_msg:
3016             \__enumext_item_answer_diff:
3017             \__enumext_log_global_vars:
3018             \__enumext_log_answer_vars:
3019             \bool_if:NTF \g__enumext_check_ans_key_bool
3020             {
3021                 \__enumext_check_ans_show:
3022             }
3023             { \__enumext_check_ans_log: }
3024             \__enumext_undefine_anskey_env:
3025         }
3026         \__enumext_reset_global_vars:
3027     }
3028 }

```

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

- This function is passed to the function `\__enumext_after_env:nn` for the environments `enumext` (§13.39) and `enumext*` (§13.44) and it is executed only when the environments are not nested or at some level of these.

### 13.33 Common functions for keyans, keyans\* and keyanspic

#### 13.33.1 Storing content in prop list

`\__enumext_keyans_addto_prop:n`

The function `\__enumext_keyans_addto_prop:n` will pass the the current `(label)` for `\item*` in `keyans` environment and the current `(label)` for `\anspic*` in `keyanspic` environment followed by the `(contents)` of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable, which will be stored to the *prop list* defined by the `save-ans` key using the function `\__enumext_store_addto_prop:V`.

```

3029 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3030 {
3031     \tl_clear:N \l__enumext_store_current_label_tl
3032     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }

```

```

3033     {
3034         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3035     }
3036     {
3037         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3038     }

```

If the *optional argument* is present and the `save-sep` key is not empty, we save it.

```

3039     \tl_if_novalue:nF { #1 }
3040     {
3041         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3042         {
3043             \tl_put_right:Ne \l__enumext_store_current_label_tl
3044             {
3045                 \l__enumext_store_keyans_item_opt_sep_tl
3046             }
3047         }
3048         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3049     }
3050     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3051 }

```

(End of definition for `\__enumext_keyans_addto_prop:n`.)

### 13.33.2 The `save-ref` key for `keyans`, `keyans*` and `keyanspic`

The “*internal label and ref*” system for the `keyans`, `keyans*` and `keyanspic` environments has *slight differences* with the one implemented for `\anskey` basically because in this environments the interest is in the current *⟨label⟩* for `\item*` and `\anspic*` with the *⟨contents⟩* of the *optional argument*. The mechanism defined here will allow to execute `\ref{⟨store name : position⟩}` and will return `1 . (A)`.

The function `\__enumext_keyans_store_ref:` handles the “*internal label and ref*” system used by the `save-ref` key for `\item*` and `\anspic*` commands. First we will create copies of the current *⟨labels⟩* and remove the dots “.” from them, we do not want to get double dots in references.

```

3052 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3053 {
3054     \bool_if:NT \l__enumext_store_ref_key_bool
3055     {
3056         \cs_set_protected:Npn \__enumext_tmp:n #1
3057         {
3058             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
3059             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
3060             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
3061             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
3062         }
3063         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3064         \__enumext_keyans_store_ref_aux_i:
3065     }
3066 }

```

The auxiliary function `\__enumext_keyans_store_ref_aux_i:` set the variable `\l__enumext_newlabel_arg_one_tl` which will contain `{⟨store name : position⟩}` analyzing whether the environment in which they are executed is `enumext*` or `enumext`.

```

3067 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3068 {
3069     \bool_if:NT \g__enumext_starred_bool
3070     {
3071         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3072     }
3073     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
3074     {
3075         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3076         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3077     }
3078     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3079     {
3080         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3081         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3082     }
3083     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
3084     {
3085         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

3086         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3087     }
3088     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3089     {
3090         \l__enumext_store_name_tl \c_colon_str
3091         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3092     }
3093     \__enumext_keyans_store_ref_aux_ii:
3094 }

```

Now auxiliary function `\__enumext_keyans_store_ref_aux_ii:` save the result in the variable `\l__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```

3095 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3096 {
3097     \tl_put_right:Ne \l__enumext_write_aux_file_tl
3098     {
3099         \__enumext_newlabel:nn
3100         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3101         { \l__enumext_newlabel_arg_two_tl }
3102     }
3103     \l__enumext_write_aux_file_tl
3104 }

```

(End of definition for `\__enumext_keyans_store_ref:`, `\__enumext_keyans_store_ref_aux_i:`, and `\__enumext_keyans_store_ref_aux_ii:`.)

### 13.33.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n`  
`\__enumext_keyans_addto_seq_link:`

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the current *⟨label⟩* `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *⟨contents⟩* of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

3105 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3106 {
3107     \tl_clear:N \l__enumext_store_current_label_tl
3108     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3109     {
3110         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3111     }
3112     {
3113         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3114     }
3115     \tl_if_novalue:nF { #1 }
3116     {
3117         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3118         {
3119             \tl_put_right:Ne \l__enumext_store_current_label_tl
3120             {
3121                 \l__enumext_store_keyans_item_opt_sep_tl
3122             }
3123         }
3124         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3125     }
3126     \__enumext_keyans_addto_seq_link:
3127 }

```

Checks if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store using the `\__enumext_store_addto_seq:V` function. Finally, copy the contents of the variable `\l__enumext_store_current_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `\__enumext_check_starred_cmd:n` and increment the value of the integer variable `\g__enumext_item_anskey_int` handled by the `check-ans` key.

```

3128 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3129 {
3130     \bool_lazy_and:nnT
3131     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3132     { \bool_if_p:N \l__enumext_hyperref_bool }
3133     {
3134         \tl_put_right:Ne \l__enumext_store_current_label_tl
3135         {
3136             \hfill \exp_not:N \hyperlink

```

```

3137         {
3138             \exp_not:V \l__enumext_newlabel_arg_one_tl
3139         }
3140         { \exp_not:V \l__enumext_mark_ref_sym_tl }
3141     }
3142 }
3143 \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3144 \bool_if:NT \l__enumext_check_answers_bool
3145 {
3146     \int_gincr:N \g__enumext_item_anskey_int
3147 }
3148 }

```

(End of definition for `\__enumext_keyans_addto_seq:n` and `\__enumext_keyans_addto_seq_link:.`)

### 13.33.4 The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off `⟨label⟩` are incorrect.

Common function to show *starred commands* `\item*` and `⟨position⟩` of stored content in *prop list* for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_⟨store name⟩_prop` for show-pos key.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:
3149 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3150 {
3151     \tl_if_novalue:nF { #1 }
3152     {
3153         \tl_set:Nc \l__enumext_store_current_opt_arg_tl { #1 }
3154     }
3155     \bool_if:NT \l__enumext_show_answer_bool
3156     {
3157         \__enumext_keyans_show_ans:
3158     }
3159     \bool_if:NT \l__enumext_show_position_bool
3160     {
3161         \__enumext_keyans_show_pos:
3162     }
3163 }
3164 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3165 {
3166     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3167     {
3168         \bool_lazy_or:nnT
3169         { \bool_if_p:N \l__enumext_show_answer_bool }
3170         { \bool_if_p:N \l__enumext_show_position_bool }
3171         {
3172             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3173         }
3174     }
3175 }
3176 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3177 {
3178     \bool_if:NT \l__enumext_starred_bool
3179     {
3180         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3181         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3182     }
3183     \tl_put_left:Nn \l__enumext_label_v_tl
3184     {
3185         \__enumext_print_keyans_box:NN
3186         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3187     }
3188 }
3189 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3190 {
3191     \bool_if:NT \l__enumext_starred_bool
3192     {
3193         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3194         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3195     }
3196     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3197     {
3198         \tl_set:Nc \l__enumext_mark_answer_sym_tl

```



```

3199     {
3200         \group_begin:
3201         \exp_not:N \normalfont
3202         \exp_not:N \footnotesize [ \int_eval:n
3203             {
3204                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3205             }
3206         ]
3207         \group_end:
3208     }
3209 }
3210 {
3211     \tl_set:Nx \l__enumext_mark_answer_sym_tl
3212     {
3213         \group_begin:
3214         \exp_not:N \normalfont
3215         \exp_not:N \footnotesize [ \int_eval:n
3216             {
3217                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3218             }
3219         ]
3220         \group_end:
3221     }
3222 }
3223 \tl_put_left:Nn \l__enumext_label_v_tl
3224 {
3225     \__enumext_print_keyans_box:NN
3226     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3227 }
3228 }

```

(End of definition for `\__enumext_keyans_show_left:n` and others.)

### 13.34 Redefining `\item` and `\makeLabel` in `enumext`

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makeLabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

When *labeling* PDF is active `\makeLabel` is redefined as `\hss #1` and the only way to get the `align` key to work correctly is to redefine `\makeLabel` using `\makebox`. The best way to implement this is to use the conditional command `\IfDocumentMetadataTF` to force this redefinition and the dedicated `mode-box` key to manually activate it by the user.

The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext` and we will add `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

`\__enumext_default_item:n`

First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `\__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `\__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```

3229 \cs_new_protected:Npn \__enumext_default_item:n #1
3230 {
3231     \tl_if_novalue:nTF {#1}
3232     {
3233         \bool_if:NT \l__enumext_check_answers_bool
3234         {
3235             \int_gincr:N \g__enumext_item_number_int
3236             \bool_set_true:N \l__enumext_item_number_bool
3237         }
3238         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3239         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3240     }
3241     {
3242         \bool_set_eq:cc
3243         { \l__enumext_wrap_label_ \__enumext_level: _bool }
3244         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3245         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3246     }
3247 }

```

(End of definition for `\__enumext_default_item:n`.)

`\__enumext_starred_item:nn`  
`\__enumext_item_star_exec:`

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` works like the *numbered* `\item`, but placing a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the value the second *optional argument* `\langle offset \rangle`.

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as “first” *optional argument* in the global variable `\g__enumext_item_symbol_aux_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item-pos*` or by the “second” *optional argument*, then we will see the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `\__enumext_item_std:w` and the key `itemindent`.

```

3248 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3249 {
3250   \tl_if_novalue:nTF {#1}
3251   {
3252     \tl_gset_eq:Nc
3253       \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3254   }
3255   {
3256     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3257   }
3258   \tl_if_novalue:nTF {#2}
3259   {
3260     \dim_set_eq:cc
3261       { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3262       { \l__enumext_labelsep_ \__enumext_level: _dim }
3263   }
3264   {
3265     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3266   }
3267   \bool_if:NT \l__enumext_check_answers_bool
3268   {
3269     \int_gincr:N \g__enumext_item_number_int
3270     \bool_set_true:N \l__enumext_item_number_bool
3271   }
3272   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3273   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3274 }

```

The function `\__enumext_item_star_exec:` will be responsible for executing `\item*` for the `enumext` environment.

```

3275 \cs_new_protected:Nn \__enumext_item_star_exec:
3276 {
3277   \tl_if_empty:cF { \l__enumext_item_symbol_ \__enumext_level: _tl }
3278   {
3279     \mode_leave_vertical:
3280     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3281     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3282     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3283   }
3284 }

```

(End of definition for `\__enumext_starred_item:nn` and `\__enumext_item_star_exec:`.)

`\__enumext_redefine_item:`

The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```

3285 \cs_new_protected:Nn \__enumext_redefine_item:
3286 {
3287   \RenewDocumentCommand \item { s o o }
3288   {
3289     \bool_if:nTF {##1}
3290     {
3291       \__enumext_starred_item:nn {##2} {##3}
3292     }
3293     { \__enumext_default_item:n {##2} }
3294   }
3295 }

```

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

`\__enumext_make_label:` The function `\__enumext_make_label:` redefine `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```

3296 \cs_new_protected:Nn \__enumext_make_label:
3297 {
3298   \IfDocumentMetadataTF
3299   {
3300     \__enumext_make_label_box:
3301   }
3302   {
3303     \bool_if:NTF \l__enumext_mode_box_bool
3304     {
3305       \__enumext_make_label_box:
3306     }
3307     {
3308       \__enumext_make_label_std:
3309     }
3310   }
3311 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3312 \cs_new_protected:Nn \__enumext_make_label_std:
3313 {
3314   \RenewDocumentCommand \makelabel { m }
3315   {
3316     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3317     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3318     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3319     {
3320       \__enumext_item_star_exec:
3321       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3322     }
3323     { ##1 }
3324     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3325     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3326   }
3327 }

```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

```

3328 \cs_new_protected:Nn \__enumext_make_label_box:
3329 {
3330   \RenewDocumentCommand \makelabel { m }
3331   {
3332     \makebox
3333     [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3334     [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3335     {
3336       \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3337       \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3338       {
3339         \__enumext_item_star_exec:
3340         \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3341       }
3342       { ##1 }
3343       \tl_gclear:N \g__enumext_item_symbol_aux_tl
3344     }
3345   }
3346 }

```

(End of definition for `\__enumext_make_label:`, `\__enumext_make_label_std:`, and `\__enumext_make_label_box:`.)

### 13.35 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` for the `enumext` and `enumext*` environments it is best to define a couple of keys that allow us to control and set by default the `\symbol` and its `\offset`.

`item-sym*` Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```

3347 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3348 {
3349   \keys_define:nn { enumext / #1 }

```

```

3350     {
3351         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
3352         item-sym* .value_required:n = true,
3353         item-sym* .initial:n = {\textasteriskcentered},
3354         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
3355         item-pos* .value_required:n = true,
3356     }
3357 }
3358 \clist_map_inline:nn
3359 {
3360     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3361 }
3362 { \__enumext_tmp:nn #1 }

```

(End of definition for `item-sym*` and `item-pos*`.)

### 13.36 Handling unknown keys

At this point in the code I already know that I will not add more *⟨keys⟩* and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the *⟨keys⟩* (you have to be consistent in life).

#### 13.36.1 Handling unknown keys for `keyans`, `keyans*` and `keyanspic`

Define and set `unknown` key for `keyans`, `keyans*` and `keyanspic` environments.

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3363 \cs_set_protected:Npn \__enumext_tmp:n #1
3364 {
3365     \keys_define:nn { enumext / #1 }
3366     {
3367         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3368     }
3369 }
3370 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3371 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3372 {
3373     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3374 }
3375 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3376 {
3377     \tl_if_blank:nTF {#2}
3378     {
3379         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3380     }
3381     {
3382         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3383     }
3384 }

```

(End of definition for `unknown`, `\__enumext_keyans_unknown_keys:n`, and `\__enumext_keyans_unknown_keys:nn`.)

#### 13.36.2 Handling unknown keys for `enumext*`

Define and set `unknown` key for `enumext*` environment.

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3385 \keys_define:nn { enumext / enumext* }
3386 {
3387     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3388 }

```

Internal functions for handling `unknown` key.

```

3389 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3390 {
3391     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3392 }
3393 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3394 {
3395     \tl_if_blank:nTF {#2}
3396     {
3397         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3398     }
3399     {
3400         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3401     }
3402 }

```

(End of definition for `unknown`, `__enumext_starred_unknown_keys:n`, and `__enumext_starred_unknown_keys:nn`.)

### 13.36.3 Handling unknown keys for enumext

`unknown` Defines and set the key `unknown` for `enumext` environment.

```

__enumext_standar_unknown_keys:n
__enumext_standar_unknown_keys:nn
3403 \cs_set_protected:Npn __enumext_tmp:n #1
3404 {
3405   \keys_define:nn { enumext / #1 }
3406   {
3407     unknown .code:n = { __enumext_standar_unknown_keys:n {##1} }
3408   }
3409 }
3410 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { __enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3411 \cs_new_protected:Npn __enumext_standar_unknown_keys:n #1
3412 {
3413   \exp_args:NV __enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3414 }
3415 \cs_new_protected:Npn __enumext_standar_unknown_keys:nn #1#2
3416 {
3417   \tl_if_blank:nTF {#2}
3418   {
3419     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3420   }
3421   {
3422     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3423   }
3424 }
```

(End of definition for `unknown`, `__enumext_standar_unknown_keys:n`, and `__enumext_standar_unknown_keys:nn`.)

### 13.37 Redefining \item and \makeLabel in keyans

The `\item` and `\item[⟨custom⟩]` commands work in the usual way in `keyans`, but the `\item*` and `\item*[⟨content⟩]` commands *store* the current *⟨label⟩* next to the *⟨content⟩* if it is present in the *sequence* and *prop list* defined by `save-ans` key.

`__enumext_keyans_default_item:n` The function `__enumext_keyans_default_item:n` executes the original behavior of the `\item` along with the keys `wrap-label`, `wrap-label*` and `itemindent`.

```

3425 \cs_new_protected:Npn __enumext_keyans_default_item:n #1
3426 {
3427   \tl_if_novalue:nTF { #1 }
3428   {
3429     \bool_set_true:N \l__enumext_wrap_label_v_bool
3430     __enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3431   }
3432   {
3433     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3434     __enumext_item_std:w [ #1 ] \tl_use:N \l__enumext_fake_item_indent_v_tl
3435   }
3436 }
```

(End of definition for `__enumext_keyans_default_item:n`.)

`__enumext_keyans_starred_item:n` The function `__enumext_keyans_starred_item:n` which will make a temporary copy of the current *⟨label⟩*, execute the `show-ans` or `show-pos` keys using the function `__enumext_keyans_show_left:n` and will display the *⟨contents⟩* of that item using the internal copy `__enumext_item_std:w`, this is necessary to prevent incrementing the current “counter” of the original *⟨label⟩*, followed by this it will execute function `__enumext_keyans_show_item_opt:` handled by `wrap-opt` key.

```

3437 \cs_new_protected:Npn __enumext_keyans_starred_item:n #1
3438 {
3439   \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3440   __enumext_keyans_show_left:n { #1 }
3441   \bool_set_true:N \l__enumext_wrap_label_v_bool
3442   __enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3443   __enumext_keyans_show_item_opt:
```

Recover the original value of the current *⟨label⟩* and *store* it first in the *prop list* (including the *optional argument*), run the internal “*label and ref*” system if the `save-ref` key is active, *store* it in the *sequence* and finally increments `g__enumext_check_starred_cmd_int` for internal check system.

```

3444   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3445   __enumext_keyans_addto_prop:n { #1 }
```

```

3446     \__enumext_keyans_store_ref:
3447     \__enumext_keyans_addto_seq:n { #1 }
3448     \int_gincr:N \__enumext_check_starred_cmd_int
3449 }

```

(End of definition for \\_\_enumext\_keyans\_starred\_item:n.)

\item\*  
 \\_\_enumext\_keyans\_redefine\_item:

The function \\_\_enumext\_keyans\_redefine\_item: is responsible for adding the *starred* argument and *optional* argument by the \\_\_enumext\_list\_arg\_two\_v: function in the definition of the *keyans* environment. Here we need to use \peek\_remove\_spaces:n to prevent an unwanted space when using \item\* in conjunction with the itemindent key. This function are passed to \\_\_enumext\_list\_arg\_two\_v: used in the definition of the *keyans* environment (§13.38.2).

```

3450 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3451 {
3452   \RenewDocumentCommand \item { s o }
3453   {
3454     \bool_if:nTF {##1}
3455     {
3456       \peek_remove_spaces:n
3457       {
3458         \__enumext_keyans_starred_item:n {##2}
3459       }
3460     }
3461     {
3462       \__enumext_keyans_default_item:n {##2}
3463     }
3464   }
3465 }

```

(End of definition for \item\* and \\_\_enumext\_keyans\_redefine\_item:. This function is documented on page 15.)

\\_\_enumext\_keyans\_make\_label:  
 \\_\_enumext\_keyans\_make\_label\_std:  
 \\_\_enumext\_keyans\_make\_label\_box:

The function \\_\_enumext\_keyans\_make\_label: redefine \makeLabel for the keys mode-box, align, font, wrap-label, wrap-label\* and \item\* for *keyans* environment. This function are passed to \\_\_enumext\_list\_arg\_two\_v: used in the definition of the *keyans* environment (§13.38.2).

```

3466 \cs_new_protected:Nn \__enumext_keyans_make_label:
3467 {
3468   \IfDocumentMetadataTF
3469   {
3470     \__enumext_keyans_make_label_box:
3471   }
3472   {
3473     \bool_if:NTF \l__enumext_mode_box_bool
3474     {
3475       \__enumext_keyans_make_label_box:
3476     }
3477     {
3478       \__enumext_keyans_make_label_std:
3479     }
3480   }
3481 }

```

Standard definition when \DocumentMetadata is not active.

```

3482 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3483 {
3484   \RenewDocumentCommand \makeLabel { m }
3485   {
3486     \tl_use:N \l__enumext_label_fill_left_v_tl
3487     \tl_use:N \l__enumext_label_font_style_v_tl
3488     \bool_if:NTF \l__enumext_wrap_label_v_bool
3489     {
3490       \__enumext_wrapper_label_v:n { ##1 }
3491     }
3492     { ##1 }
3493     \tl_use:N \l__enumext_label_fill_right_v_tl
3494   }
3495 }

```

Definition using \makebox when \DocumentMetadata is active or mode-box is active.

```

3496 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3497 {
3498   \RenewDocumentCommand \makeLabel { m }

```

```

3499 {
3500   \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3501   {
3502     \tl_use:N \l__enumext_label_font_style_v_tl
3503     \bool_if:NTF \l__enumext_wrap_label_v_bool
3504     {
3505       \__enumext_wrapper_label_v:n { ##1 }
3506     }
3507     { ##1 }
3508   }
3509 }
3510 }

```

(End of definition for `\__enumext_keyans_make_label:`, `\__enumext_keyans_make_label_std:`, and `\__enumext_keyans_make_label_box:`.)

### 13.38 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `\__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

#### 13.38.1 Calculation of `\leftmargin` and `\itemindent`

Consider the figure 9 where the default margins (on the left) of a list are represented.

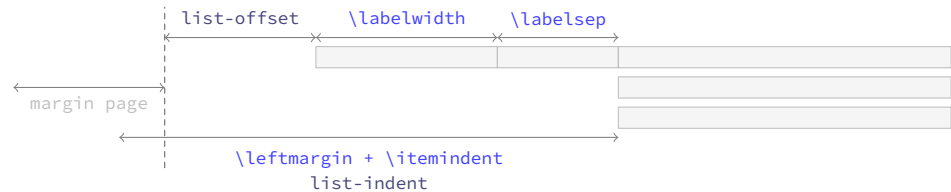


Figure 9: Representation of standard horizontal lengths in `list` environment.

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The key relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the `label box` is at `\leftmargin + \itemindent` minus `\labelwidth + \labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

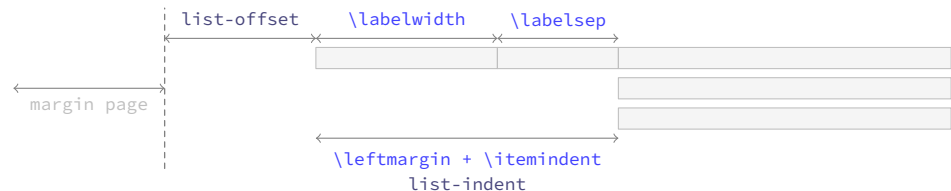


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

Where the default values will look like in the figure 11.

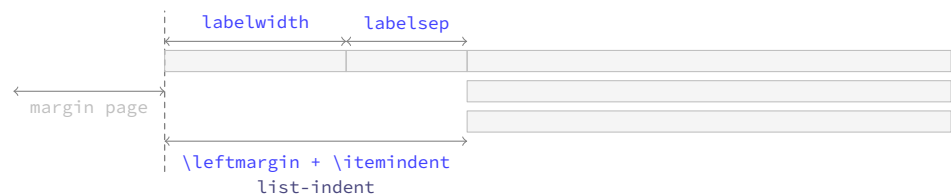


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

The function `\__enumext_calc_hspace:NNNNNNN` takes seven arguments to be able to determine horizontal spaces for all list environment:

#1: <code>\l__enumext_labelwidth_X_dim</code>	#2: <code>\l__enumext_labelsep_X_dim</code>
#3: <code>\l__enumext_listoffset_X_dim</code>	#4: <code>\l__enumext_leftmargin_tmp_X_dim</code>
#5: <code>\l__enumext_leftmargin_X_dim</code>	#6: <code>\l__enumext_itemindent_X_dim</code>
#7: <code>\l__enumext_leftmargin_tmp_X_bool</code>	

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§13.38.2).

```

3511 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3512 {
3513   \dim_compare:nNt { #1 } < { \c_zero_dim }

```



```

3514     {
3515         \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3516         \dim_set:Nn #1 { \dim_abs:n { #1 } }
3517     }
3518     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3519     {
3520         \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3521         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3522     }

```

If no value has been passed to the `labelwidth` and `labelsep` keys we set the default values for `\l__enumext_leftmargin_tmp_X_dim`.

```

3523     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

We now analyze the cases and set the values for `\leftmargin` and `\itemindent`.

```

3524     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3525     {
3526         \dim_set:Nn #6 { #1 + #2 - #4 }
3527         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3528     }
3529     {
3530         \dim_compare:nNnT { #4 } = { #1 + #2 }
3531         { \dim_set:Nn #6 { \c_zero_dim } }
3532         \dim_compare:nNnT { #4 } < { #1 + #2 }
3533         { \dim_set:Nn #6 { #1 + #2 - #4 } }
3534         \dim_compare:nNnT { #4 } > { #1 + #2 }
3535         {
3536             \dim_set:Nn #6 { -#1 - #2 + #4 }
3537             \dim_set:Nn #6 { #6*-1 }
3538         }
3539         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3540     }
3541 }
3542 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(End of definition for `\__enumext_calc_hspace:NNNNNNN`.)

### 13.38.2 Setting second argument of the lists

We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3543 \cs_set_protected:Npn \__enumext_tmp:n #1
3544 {
3545     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3546     {
3547         \__enumext_calc_hspace:ccccc
3548         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3549         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3550         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3551         { \__enumext_leftmargin_tmp_#1_bool }
3552         \clist_map_inline:nn
3553         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3554         { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3555         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3556         { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3557         \usecounter { enumX#1 }
3558         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3559         \str_if_eq:nnTF {#1} { v }
3560         {
3561             \__enumext_keyans_redefine_item:
3562             \__enumext_keyans_make_label:
3563             \__enumext_keyans_ref:
3564             \__enumext_keyans_fake_item_indent:
3565             \bool_if:cT { \__enumext_show_length_#1_bool }
3566             {
3567                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3568             }
3569         }
3570         {
3571             \__enumext_redefine_item:
3572             \__enumext_make_label:
3573             \__enumext_standar_ref:

```

```

3574         \__enumext_fake_item_indent:
3575         \bool_if:cT { \__enumext_show_length_#1_bool }
3576         {
3577             \msg_term:nnne { enumext } { list-lengths } {#1}
3578             { \int_use:N \__enumext_level_int }
3579         }
3580     }
3581 }
3582 }
3583 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for \\_\_enumext\_list\_arg\_two\_i: and others.)

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `0pt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `listparindent` to make it set the value of `\parskip` locally.

```

3584 \cs_set_protected:Npn \__enumext_tmp:n #1
3585 {
3586     \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
3587     {
3588         \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3589         \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3590         \__enumext_calc_hspace:cccccc
3591         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3592         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3593         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3594         { \__enumext_leftmargin_tmp_#1_bool }
3595         \clist_map_inline:nn
3596         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3597         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3598         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3599         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3600         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3601         \skip_zero:N \partopsep
3602         \usecounter { enumX#1 }
3603         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3604         \__enumext_starred_ref:
3605         \str_if_eq:nnTF {#1} { vii }
3606         {
3607             \__enumext_fake_item_indent_vii:
3608             \bool_if:cT { \__enumext_show_length_vii_bool }
3609             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3610         }
3611         {
3612             \__enumext_fake_item_indent_viii:
3613             \bool_if:cT { \__enumext_show_length_#1_bool }
3614             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3615         }
3616     }
3617 }
3618 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for \\_\_enumext\_list\_arg\_two\_vii: and \\_\_enumext\_list\_arg\_two\_viii:.)

### 13.39 The environment enumext

```
\__enumext_safe_exec:
```

The `\__enumext_safe_exec:` function first call the function `\__enumext_is_not_nested:` which sets `\g__enumext_standar_bool` to “true” if we are NOT nested within `enumext*`, then call the function `\__enumext_internal_mini_page:` to create the environment `\__enumext_mini_page`, we will increment `\l__enumext_level_int` to restrict nesting of the environment, set `\l__enumext_standar_bool` to “true” and finally call the function `\__enumext_is_on_first_level:` which sets `\l__enumext_standar_first_bool` to “true” only if the environment is NOT nested and we are at the “first level”.

```

3619 \cs_new_protected:Nn \__enumext_safe_exec:
3620 {
3621     \__enumext_is_not_nested:
3622     \__enumext_internal_mini_page:
3623     \int_incr:N \l__enumext_level_int
3624     \int_compare:nNt { \l__enumext_level_int } > { 4 }
3625     { \msg_fatal:nn { enumext } { list-too-deep } }
3626     \bool_set_true:N \l__enumext_standar_bool

```

```

3627     \bool_set_false:N \l__enumext_starred_bool
3628     \__enumext_is_on_first_level:
3629 }

```

(End of definition for \\_\_enumext\_safe\_exec:.)

\\_\_enumext\_parse\_keys:n

The \\_\_enumext\_parse\_keys:n function first we will clear the variable \l\_\_enumext\_series\_str used by the key `series` and then we check if we are at the “*first level*”, if so we process the *⟨keys⟩* and then execute the function \\_\_enumext\_parse\_series:n used by the key `series` and call the function \\_\_enumext\_nested\_base\_line\_fix: used by the key `base-fix`, otherwise we will pass the *⟨keys⟩* to the inner levels of the environment then we execute the function \\_\_enumext\_store\_active\_keys:n and reprocess the *⟨keys⟩* to pass them to the *sequence* if the key `save-key` is not active.

```

3630 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3631 {
3632     \tl_if_novalue:nF {#1}
3633     {
3634         \str_clear:N \l__enumext_series_str
3635         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3636         {
3637             \keys_set:nn { enumext / level-1 } {#1}
3638             \__enumext_parse_series:n {#1}
3639             \__enumext_nested_base_line_fix:
3640         }
3641         {
3642             \exp_args:Ne \keys_set:nn
3643             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3644         }
3645         \__enumext_store_active_keys:n {#1}
3646     }
3647 }

```

(End of definition for \\_\_enumext\_parse\_keys:n.)

\\_\_enumext\_start\_store\_level:

The \\_\_enumext\_start\_store\_level: function activate the “*storing structure*” mechanism in the *sequence* for the command \anskey and the environment `anskey*`.

```

3648 \cs_new_protected:Nn \__enumext_start_store_level:
3649 {
3650     \bool_lazy_all:nT
3651     {
3652         { \bool_if_p:N \l__enumext_store_active_bool }
3653         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3654         { \bool_if_p:N \g__enumext_standar_bool }
3655     }
3656     {
3657         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3658         {
3659             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3660             \__enumext_store_level_open:
3661         }
3662     }

```

If `enumext` are nested in `enumext*` add \\_\_enumext\_store\_level\_open: to preserve the “*storing structure*”.

```

3663     \bool_lazy_all:nT
3664     {
3665         { \bool_if_p:N \l__enumext_store_active_bool }
3666         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3667         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3668     }
3669     {
3670         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3671         {
3672             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3673             \__enumext_store_level_open:
3674         }
3675     }
3676 }

```

(End of definition for \\_\_enumext\_start\_store\_level:.)

`\__enumext_stop_store_level:` The `\__enumext_stop_store_level:` function stop the “*storing structure*” mechanism in the *sequence* for the command `\anskey` and the environment `anskey*`.

```

3677 \cs_new_protected:Nn \__enumext_stop_store_level:
3678 {
3679   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3680   {
3681     \__enumext_store_level_close:
3682   }
3683 }
```

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

`\__enumext_multicols_start:` The function `\__enumext_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key, then set the default value for `\columnsep` when `columns-sep=opt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

3684 \cs_new_protected:Nn \__enumext_multicols_start:
3685 {
3686   \int_compare:nNt
3687   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3688   {
3689     \dim_compare:nNt
3690     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3691     {
3692       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3693       {
3694         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3695         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3696         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3697         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3698       }
3699     }
3700     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3701     \int_compare:nNt { \__enumext_level_int } > { 1 }
3702     {
3703       \dim_zero:N \columnseprule
3704     }
3705 }
```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `\__enumext_multi_addvspace:`, apply our “*vertical adjust spacing*”, then start the `multicols` environment.

```

3705   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3706   {
3707     \skip_zero:N \multicolsep
3708     \__enumext_multi_addvspace:
3709   }
3710   \raggedcolumns
3711   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3712 }
3713 }
```

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

`\__enumext_multicols_stop:` The function `\__enumext_multicols_stop:` will stop the `multicols` environment and apply our “*vertical adjust*” spacing. For compatibility with *tagged PDF*, the closing of the `list` environment is executed here along with `\__enumext_stop_store_level:`.

```

3714 \cs_new_protected:Nn \__enumext_multicols_stop:
3715 {
3716   \int_compare:nNtF
3717   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3718   {
3719     \__enumext_stop_list:
3720     \__enumext_stop_store_level:
3721     \end{multicols}
3722     \__enumext_unskip_unkern:
3723     \__enumext_unskip_unkern:
3724     \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3725   }
3726   {
3727     \__enumext_stop_list:
3728     \__enumext_stop_store_level:
3729   }
3730 }
```

(End of definition for `\__enumext_multicols_stop:`)

`\__enumext_before_list:` The function `\__enumext_before_list:` first calls the function `\__enumext_vspace_above:` used by the keys `above` and `above*`, then calls the function `\__enumext_before_args_exec:` used by the key `before*` and finally execute the function `\__enumext_check_ans_active:` for the check answer mechanism.

```

3731 \cs_new_protected:Nn \__enumext_before_list:
3732 {
3733   \__enumext_vspace_above:
3734   \__enumext_before_args_exec:
3735   \__enumext_check_ans_active:

```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_X_dim` to be the *width* of the `\__enumext_mini_page` environment on the “*right side*”, using this value together with the value of the `\l__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_X_dim` will be set, which will be the *width* of `\__enumext_mini_page` environment on the “*left side*”, always having a current `\linewidth` as *maximum width* between them.

```

3736   \dim_compare:nNt
3737   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3738   {
3739     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3740     {
3741       \linewidth
3742       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3743       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3744     }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `\__enumext_minipage_add_space:` is called and the `\__enumext_mini_page` environment on the “*left side*” will be initialized followed by the “*vertical spacing*” applied to preserve the “*baseline*” between the *left* and *right* side environments. After these actions, the function `\__enumext_multicols_start:` is called to handle the `multicols` environment.

```

3745   \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3746   \int_gincr:N \g__enumext_minipage_stat_int
3747   \__enumext_minipage_add_space:
3748   \noindent
3749   \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3750 }
3751 \__enumext_multicols_start:
3752 }

```

(End of definition for `\__enumext_before_list:`)

`\__enumext_second_part:` The function `\__enumext_second_part:` first check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is “*true*” a small test will be executed to check if we have omitted the use of `\miniright` (the `\__enumext_mini_page` environment has not been closed), then close `\__enumext_mini_page` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3753 \cs_new_protected:Nn \__enumext_second_part:
3754 {
3755   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3756   {
3757     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3758     {
3759       \msg_warning:nn { enumext } { missing-miniright }
3760       \miniright
3761     }
3762     \int_gzero:N \g__enumext_minipage_stat_int
3763     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3764     \end__enumext_mini_page
3765   }
3766   {
3767     \__enumext_multicols_stop:
3768   }

```

Now we will execute the functions `\__enumext_after_stop_list:` used by the key `after`, `\__enumext_check_ans_key_hook:` used by the key `check-ans`, `\__enumext_vspace_below:` used by the keys `below` and `below*`. Finally set `\l__enumext_standar_bool` to false and call the function `\__enumext_resume_save_counter:` used by the `series`, `resume` and `resume*` keys.

```

3769   \__enumext_after_stop_list:
3770   \__enumext_check_ans_key_hook:

```

```

3771     \__enumext_vspace_below:
3772     \bool_set_false:N \__enumext_standar_bool
3773     \__enumext_resume_save_counter:
3774 }

```

(End of definition for \\_\_enumext\_second\_part:.)

\\_\_enumext\_set\_item\_width: The function \\_\_enumext\_set\_item\_width: will set the value of \itemwidth taking into account the value established by the list-offset key for each level of the environment.

```

3775 \cs_new_protected:Nn \__enumext_set_item_width:
3776 {
3777     \dim_set:Nn \itemwidth { \linewidth }
3778     \dim_compare:nT
3779     {
3780         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3781     }
3782     {
3783         \dim_sub:Nn \itemwidth
3784         {
3785             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3786         }
3787     }
3788 }

```

(End of definition for \\_\_enumext\_set\_item\_width:.)

**enumext** Now create the enumext environment based on list environment by levels.

```

3789 \NewDocumentEnvironment{enumext}{0}{}
3790 {
3791     \__enumext_safe_exec:
3792     \__enumext_parse_keys:n {#1}
3793     \__enumext_before_list:
3794     \__enumext_start_store_level:
3795     \__enumext_start_list:nn
3796     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3797     {
3798         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3799         \__enumext_before_keys_exec:
3800     }
3801     \__enumext_set_item_width:
3802     \__enumext_after_args_exec:
3803 }
3804 {
3805     \__enumext_second_part:
3806 }

```

(End of definition for enumext. This function is documented on page 5.)

As we don't want our check to be executed check-ans by levels but on the complete list, we will take it out of the enumext environment using the "hook" function \\_\_enumext\_after\_env:nn.

```

3807 \__enumext_after_env:nn {enumext}
3808 {
3809     \__enumext_execute_after_env:
3810 }

```

### 13.40 The environment keyans

The environment keyans also based on lists. The main differences with the enumext environment are the nesting and the way the answers (choice) will be stored and checked, this environment is intended exclusively for "multiple choice questions".

\\_\_enumext\_keyans\_safe\_exec: The keyans environment will only be available if the save-ans key is active and can only be used at the "first level" within the enumext environment. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```

3811 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3812 {
3813     \bool_if:NF \__enumext_store_active_bool
3814     {
3815         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3816     }
3817     \int_incr:N \__enumext_keyans_level_int
3818     \bool_set_true:N \__enumext_keyans_env_bool

```

```

3819 \__enumext_keyans_name_and_start:
3820 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3821 \bool_set_false:N \__enumext_store_active_bool
3822 \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
3823 {
3824   \msg_error:nn { enumext } { keyans-nested }
3825 }
3826 \int_compare:nNnT { \__enumext_level_int } > { 1 }
3827 {
3828   \msg_error:nn { enumext } { keyans-wrong-level }
3829 }
3830 }

```

(End of definition for \\_\_enumext\_keyans\_safe\_exec:.)

```

\__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
3831 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3832 {
3833   \keys_set:nn { enumext / keyans } {#1}
3834 }

```

(End of definition for \\_\_enumext\_keyans\_parse\_keys:n.)

\\_\_enumext\_before\_list\_v: Same implementation as the one used in the enumext environment.

```

\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_second_part_v:
3835 \cs_new_protected:Nn \__enumext_before_list_v:
3836 {
3837   \__enumext_vspace_above_v:
3838   \__enumext_before_args_exec_v:
3839   \dim_compare:nNnT { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
3840   {
3841     \dim_set:Nn \__enumext_minipage_left_v_dim
3842     {
3843       \linewidth - \__enumext_minipage_right_v_dim - \__enumext_minipage_hsep_v_dim
3844     }
3845     \bool_set_true:N \__enumext_minipage_active_v_bool
3846     \int_gincr:N \g__enumext_minipage_stat_int
3847     \__enumext_keyans_minipage_add_space:
3848     \__enumext_mini_page{ \__enumext_minipage_left_v_dim }
3849   }
3850   \__enumext_keyans_multicols_start:
3851 }
3852 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3853 {
3854   \int_compare:nNnT { \__enumext_columns_v_int } > { 1 }
3855   {
3856     \dim_compare:nNnT { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3857     {
3858       \dim_set:Nn \__enumext_columns_sep_v_dim
3859       {
3860         (
3861           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3862         ) / \__enumext_columns_v_int
3863         - \__enumext_listoffset_v_dim
3864       }
3865     }
3866     \dim_set_eq:NN \columnsep \__enumext_columns_sep_v_dim
3867     \dim_zero:N \columnseprule % no rule here
3868     \bool_if:NF \__enumext_minipage_active_v_bool
3869     {
3870       \skip_zero:N \multicolsep
3871       \__enumext_keyans_multi_addvspace:
3872     }
3873     \raggedcolumns
3874     \begin{multicols}{ \__enumext_columns_v_int }
3875   }
3876 }
3877 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3878 {
3879   \int_compare:nNnTF { \__enumext_columns_v_int } > { 1 }
3880   {
3881     \__enumext_stop_list:

```



```

3882         \end{multicols}
3883         \__enumext_unskip_unkern:
3884         \__enumext_unskip_unkern:
3885         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3886     }
3887     {
3888         \__enumext_stop_list:
3889     }
3890 }
3891 \cs_new_protected:Nn \__enumext_second_part_v:
3892 {
3893     \bool_if:NTF \l__enumext_minipage_active_v_bool
3894     {
3895         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3896         {
3897             \msg_warning:nn { enumext } { missing-miniright }
3898             \miniright
3899         }
3900         \int_gzero:N \g__enumext_minipage_stat_int
3901         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3902         \end__enumext_mini_page
3903         \par\addvspace{ \l__enumext_minipage_after_skip }
3904     }
3905     {
3906         \__enumext_keyans_multicols_stop:
3907     }
3908     \bool_set_false:N \l__enumext_keyans_env_bool
3909     \__enumext_after_stop_list_v:
3910     \__enumext_vspace_below_v:
3911 }

```

(End of definition for \\_\_enumext\_before\_list\_v: and others.)

\\_\_enumext\_keyans\_set\_item\_width:

The function \\_\_enumext\_keyans\_set\_item\_width: will set the value of \itemwidth taking into account the value established by the list-offset key.

```

3912 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3913 {
3914     \dim_set:Nn \itemwidth { \linewidth }
3915     \dim_compare:nT
3916     {
3917         \l__enumext_listoffset_v_dim != \c_zero_dim
3918     }
3919     {
3920         \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3921     }
3922 }

```

(End of definition for \\_\_enumext\_keyans\_set\_item\_width:.)

keyans

Now we define the environment `keyans` also based on lists.

```

3923 \NewDocumentEnvironment{keyans}{ 0{} }
3924 {
3925     \__enumext_keyans_safe_exec:
3926     \__enumext_keyans_parse_keys:n {#1}
3927     \__enumext_before_list_v:
3928     \__enumext_start_list:nn
3929     { \tl_use:N \l__enumext_label_v_tl }
3930     {
3931         \__enumext_list_arg_two_v:
3932         \__enumext_before_keys_exec_v:
3933     }
3934     \__enumext_keyans_set_item_width:
3935     \__enumext_after_args_exec_v:
3936 }
3937 {
3938     \__enumext_check_starred_cmd:n { item }
3939     \__enumext_second_part_v:
3940 }

```

(End of definition for `keyans`. This function is documented on page 15.)

### 13.41 Tagging PDF support for non-standart list environments

The  $\TeX$  release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a `list` environment is a `list` environment, no matter what it looks like in the PDF file.

To maintain a correct `list` structure when `\DocumentMetadata` is active, it is necessary to do some things manually. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in [How can I modify my \item redefinition to be compatible with tagging-pdf](#).

#### 13.41.1 Socket for tagging support in enumext\* and keyans\*

We will first define the necessary sockets and their behavior for `enumext*` and `keyans*`.

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
  __enumext_stop_start_list_tag:
__enumext_stop_list_tag:n
3941 \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3942 \socket_new_plugin:nnn {tagsupport/enumext/starred} {start-list-tags}
3943 {
3944   \tag_resume:n {#1}
3945   \tag_struct_begin:n {tag=LI}
3946   \tag_struct_begin:n {tag=Lbl}
3947   \tag_mc_begin:n {tag=Lbl}
3948 }
3949 \socket_new_plugin:nnn {tagsupport/enumext/starred} {stop-start-tags}
3950 {
3951   \tag_mc_end:
3952   \tag_struct_end:n {tag=Lbl}
3953   \tag_struct_begin:n {tag=LBody}
3954   \tag_struct_begin:n {tag=text-unit}
3955   \tag_struct_begin:n {tag=text}
3956 }
3957 \socket_new_plugin:nnn {tagsupport/enumext/starred} {stop-list-tags}
3958 {
3959   \tag_struct_end:n {tag=text}
3960   \tag_struct_end:n {tag=text-unit}
3961   \tag_struct_end:n {tag=LBody}
3962   \tag_struct_end:n {tag=LI}
3963   \tag_suspend:n {#1}
3964 }
```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

3965 \cs_new_protected_nopar:Npn __enumext_start_list_tag:n #1
3966 {
3967   \IfDocumentMetadataTF
3968   {
3969     \socket_assign_plugin:nn {tagsupport/enumext/starred} {start-list-tags}
3970     \socket_use:n {tagsupport/enumext/starred} {#1}
3971   } {}
3972 }
3973 \cs_new_protected_nopar:Nn __enumext_stop_start_list_tag:
3974 {
3975   \IfDocumentMetadataTF
3976   {
3977     \socket_assign_plugin:nn {tagsupport/enumext/starred} {stop-start-tags}
3978     \socket_use:nn {tagsupport/enumext/starred} { }
3979   } {}
3980 }
3981 \cs_new_protected_nopar:Npn __enumext_stop_list_tag:n #1
3982 {
3983   \IfDocumentMetadataTF
3984   {
3985     \socket_assign_plugin:nn {tagsupport/enumext/starred} {stop-list-tags}
3986     \socket_use:nn {tagsupport/enumext/starred} {#1}
3987   } {}
3988 }
```

(End of definition for *start-list-tags* and others.)

#### 13.41.2 Socket for tagging support in keyanspic

We will first define the necessary sockets and their behavior for `keyanspic` environment.

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:
3989 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3990 \socket_new_plugin:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3991 {
```

```
3992 \tag_resume:n {keyanspic}
3993 \tag_struct_begin:n {tag=LI}
3994 \tag_struct_begin:n {tag=Lbl}
3995 \tag_mc_begin:n {tag=Lbl}
3996 }
3997 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3998 {
3999 \tag_mc_end:
4000 \tag_struct_end:n {tag=Lbl}
4001 \tag_struct_begin:n {tag=LBody}
4002 \tag_struct_begin:n {tag=text-unit}
4003 \tag_struct_begin:n {tag=text}
4004 \tag_mc_begin:n {tag=text}
4005 }
4006 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
4007 {
4008 \tag_mc_end:
4009 \tag_struct_end:n {tag=text-unit}
4010 \tag_struct_end:n {tag=text}
4011 \tag_struct_end:n {tag=LBody}
4012 \tag_struct_end:n {tag=LI}
4013 \tag_suspend:n {keyanspic}
4014 }
```

And now we'll wrap them so that they're only active when \DocumentMetadata is present.

```
4015 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4016 {
4017 \IfDocumentMetadataTF
4018 {
4019 \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
4020 \socket_use:n {tagsupport/enumext/keyanspic}
4021 } {}
4022 }
4023 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4024 {
4025 \IfDocumentMetadataTF
4026 {
4027 \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
4028 \socket_use:nn {tagsupport/enumext/keyanspic}
4029 } {}
4030 }
4031 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4032 {
4033 \IfDocumentMetadataTF
4034 {
4035 \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
4036 \socket_use:nn {tagsupport/enumext/keyanspic}
4037 } {}
4038 }
```

(End of definition for start-list-tags and others.)

13.42 The environment keyanspic and \anspic

The `keyanspic` environment is a `list` based environment that uses the same configuration for “spacing” and `<label>` as the `keyans` environment, but it does not use `\item`. The `<contents>` are passed to the environment by means of the `\anspic` command as replacement for `\item` command and placed inside `minipage` environments, with the `<label>` centered “above” or “below”, adjusting *widths* and *position* according to the options passed to the environment.

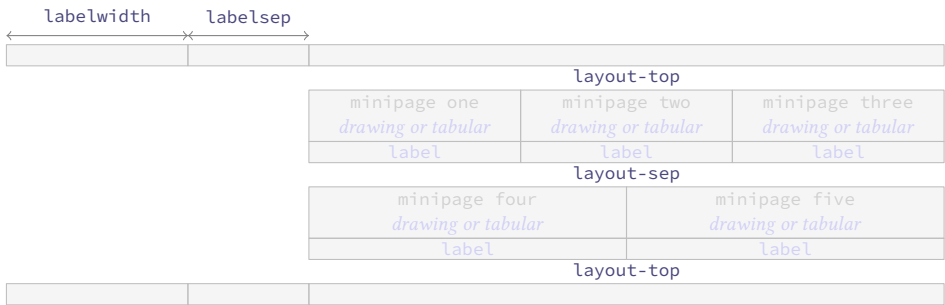


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a \macro?](#).

### 13.42.1 The environment `keyanspic`

First we define the key that allows us to process the position of the `\label` centered “above” or “below” which will be `label-pos`, the vertical separation of these from *drawing* or *tabular* will be handled with the key `label-sep`. The “*layout style*” will be handled with the key `layout-sty` will take two values separated by comma `{\n° upper, n° lower}` and will determine the number of `minipage` environments in which all arguments of `\anspic` will be printed at the “upper” and “lower” within the environments separated by the value of the key `layout-sep`. The vertical space “top” and “bottom” of the environment will be handled with the key `layout-top`.

```

4039 \keys_define:nn { enumext / keyanspic }
4040 {
4041   label-pos .choice:,
4042   label-pos / above .code:n =
4043     \bool_set_true:N \l__enumext_anspic_label_above_bool
4044     \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4045   label-pos / below .code:n =
4046     \bool_set_false:N \l__enumext_anspic_label_above_bool
4047     \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4048   label-pos / unknown .code:n =
4049     \msg_error:nneee { enumext } { unknown-choice }
4050     { label-pos } { above,~ below } { \exp_not:n {#1} },
4051   label-pos .initial:n = below,
4052   label-pos .value_required:n = true,
4053   label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4054   label-sep .value_required:n = true,
4055   layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4056   layout-sty .value_required:n = true,
4057   layout-sep .code:n = \keys_set:nn { enumext / keyans }
4058     { parsep = #1 },
4059   layout-sep .value_required:n = true,
4060   layout-top .code:n = \keys_set:nn { enumext / keyans }
4061     { topsep = #1 },
4062   layout-top .value_required:n = true,
4063   unknown .code:n = { \__enumext_keyans_unknown_keys:n {#1} }
4064 }

```

(End of definition for `label-pos` and others.)

```

\__enumext_keyans_pic_safe_exec:
\__enumext_keyans_pic_parse_keys:n
\__enumext_keyans_pic_skip_abs:N
\__enumext_keyans_pic_arg_two:

```

The function `\__enumext_keyans_pic_safe_exec`: check the nested level position inside the `enumext` environment.

```

4065 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4066 {
4067   \int_incr:N \l__enumext_keyans_pic_level_int
4068   \int_compare:nNtT { \l__enumext_keyans_pic_level_int } > { 1 }
4069   {
4070     \msg_error:nn { enumext } { keyanspic-nested }
4071   }
4072   \__enumext_keyans_name_and_start:
4073 }

```

Parse `[<key = val>]` for `keyanspic` environment.

```

4074 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4075 {
4076   \tl_if_no_value:nF {#1}
4077   {
4078     \keys_set:nn { enumext / keyanspic } {#1}
4079   }
4080 }

```

The function `\__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep` from `keyans` environment.

```

4081 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4082 {
4083   \dim_compare:nNtT { #1 } < { \c_zero_dim }
4084   {
4085     \skip_set:Nn #1 { -#1 }
4086   }
4087 }

```

The `\__enumext_keyans_pic_arg_two:` function will be used in the *second argument* of the `list` environment that defines the `keyanspic` environment, with this we will take the configuration of the “spaces” and the keys `label`, `wrap-label`, `parsep` and `topsep` from the `keyans` environment. The first thing we need to do is set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to “false”, then copy the definition of the second list argument from the `keyans` environment definition and make sure that `\parsep` does not have a negative value.

```
4088 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4089 {
4090   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4091   \__enumext_list_arg_two_v:
4092   \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the counter `enumXv` of the `keyans` environment and save the *total height* of the `(label)` in `\l__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the key `label-pos` is set to *below*.

```
4093   \bool_if:NF \l__enumext_anspic_label_above_bool
4094   {
4095     \stepcounter { enumXv }
4096     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4097     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4098     {
4099       \box_ht_plus_dp:N \l__enumext_anspic_label_box
4100     }
4101     \skip_add:Nn \parsep
4102     {
4103       \l__enumext_anspic_label_htdp_dim
4104       + \box_dp:N \strutbox
4105       + \l__enumext_anspic_label_sep_skip
4106     }
4107   }
```

Finally we *adjust* the value of `\leftmargin` and `\topsep` then set `\listparindent`, `\partopsep` and `\itemsep` to zero so that the *horizontal* and *vertical* space is not affected.

```
4108   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
4109   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4110   \dim_zero:N \listparindent
4111   \skip_zero:N \partopsep
4112   \skip_zero:N \itemsep
4113 }
```

(End of definition for `\__enumext_keyans_pic_safe_exec:` and others.)

`keyanspic` Now we define the environment `keyanspic`. For compatibility with *tagged* PDF we must use the `\begin{list}` form and a lot of conditional code using `\IfDocumentMetadataTF`.

```
4114 \NewDocumentEnvironment{keyanspic}{o}
4115 {
4116   \__enumext_keyans_pic_safe_exec:
4117   \__enumext_keyans_pic_parse_keys:n {#1}
4118   \begin{list} {} { \__enumext_keyans_pic_arg_two: }
4119   \IfDocumentMetadataTF
4120   {
4121     \tag_suspend:n {list}
4122   }{}
4123   \item[] \scan_stop:
4124   % paranoia
4125   \RenewDocumentCommand \item {}
4126   {
4127     \msg_error:nn { enumext } { keyanspic-item-cmd }
4128   }
4129   \IfDocumentMetadataTF
4130   {
4131     \tag_resume:n {keyanspic}
4132     \tag_tool:n {para/tagging=false}
4133     \tag_suspend:n {keyanspic}
4134   } {}
4135 }
4136 {
4137   \IfDocumentMetadataTF
4138   {
4139     \tag_resume:n {keyanspic}
4140     \tag_struct_begin:n {tag=L,attribute=enumerate}
4141   } {} }
```

Now we process the command `\anspic`, if the key `layout-sty` is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_anspic_args_seq` and placed a *single line*.

```

4142   \__enumext_anspic_exec:
4143   \IfDocumentMetadataTF
4144   {
4145     \tag_suspend:n {keyanspic}
4146   } { }
4147   \end{list}
4148   \IfDocumentMetadataTF
4149   {
4150     \tag_struct_end:
4151     \tag_struct_end:
4152   } { }

```

Finally we check if `\anspic*` has been used, set the counter `enumXvi` to zero and apply our “adjusted” vertical space bottom.

```

4153   \__enumext_check_starred_cmd:n { anspic }
4154   \setcounter { enumXvi } { 0 }
4155   \bool_if:NTF \l__enumext_anspic_label_above_bool
4156   {
4157     \par\addvspace{ 0.5\box_dp:N \strutbox }
4158   }
4159   {
4160     \par
4161     \addvspace
4162     {
4163       \dim_eval:n
4164       {
4165         \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4166         + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4167       }
4168     }
4169   }
4170 }

```

(End of definition for `keyanspic`. This function is documented on page 16.)

### 13.42.2 The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*[\langle content \rangle]` store the current `\label` next to the *optional argument* `[\langle content \rangle]` in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* `{\langle drawing or tabular \rangle}` is NOT stored in the *sequence* or *prop list*.

- One of the complications here to make the `keyanspic` environment compatible with *tagged PDF* is the position of `\label`, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to `\label` and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is `\label`, is above #3 there are no problems with *tagged PDF*, but if #3 comes first the list created with *tagged PDF* will not be correct.

`\anspic`

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error. The three arguments are handled by the function `\__enumext_anspic_args:nnn` and stored in the sequence `\l__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

4171 \NewDocumentCommand \anspic { s o +m }
4172 {
4173   \bool_if:NF \l__enumext_store_active_bool
4174   {
4175     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4176   }
4177   \int_compare:nNt { \l__enumext_level_int } > { 1 }
4178   {
4179     \msg_error:nn { enumext } { keyanspic-wrong-level }
4180   }
4181   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
4182   {
4183     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4184   }
4185   \seq_put_right:Nn \l__enumext_anspic_args_seq
4186   {
4187     \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4188   }
4189 }

```

The `\__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the “height plus depth” of the *mandatory argument* if the key `label-pos` is set “below”.

```

4190 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4191 {
4192   \bool_if:NF \l__enumext_anspic_label_above_bool
4193   {
4194     \IfDocumentMetadataTF
4195     {
4196       \tag_suspend:n {keyanspic}
4197     } { }
4198     \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4199     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4200     {
4201       \box_ht_plus_dp:N \l__enumext_anspic_body_box
4202     }
4203     \IfDocumentMetadataTF
4204     {
4205       \tag_resume:n {keyanspic}
4206     } { }
4207   }
4208 }

```

The `\__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* ‘\*’ and *optional argument* passed to the command. Here we will store the `<label>` and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label` and `wrap-opt` keys.

```

4209 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4210 {
4211   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4212   {
4213     \bool_if:nT { #1 }
4214     {
4215       \__enumext_keyans_addto_prop:n { #2 }
4216       \__enumext_keyans_store_ref:
4217       \__enumext_keyans_addto_seq:n { #2 }
4218       \int_gincr:N \g__enumext_check_starred_cmd_int
4219       \bool_lazy_or:nnT
4220       { \bool_if_p:N \l__enumext_show_answer_bool }
4221       { \bool_if_p:N \l__enumext_show_position_bool }
4222       {
4223         \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4224         \__enumext_keyans_show_left:n { #2 }
4225         \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4226       }
4227     }
4228     \tl_use:N \l__enumext_label_font_style_v_tl
4229     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4230     \__enumext_keyans_show_item_opt:
4231   }
4232 }

```

The function `\__enumext_anspic_label_pos:nnn` will be in charge of handling the “counter” and the position of the `<label>`, set by `label-pos` key which will have the same configuration as the `keyans` environment.

```

4233 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4234 {
4235   \stepcounter { enumXvi }
4236   \__enumext_anspic_body_dim:n { #3 }
4237   \bool_if:NTF \l__enumext_anspic_label_above_bool
4238   {
4239     \__enumext_anspic_label:nn { #1 } { #2 }
4240   }
4241   {
4242     \raisebox
4243     {
4244       -\dim_eval:n
4245       {
4246         \l__enumext_anspic_label_htdp_dim
4247         + \l__enumext_anspic_body_htdp_dim
4248         + \box_dp:N \strutbox
4249         + \l__enumext_anspic_label_sep_skip
4250       }
4251     }

```



```

4252         [ Opt ] [ Opt ]
4253     {
4254         \__enumext_anspic_label:nn { #1 } { #2 }
4255     }
4256 }
4257 }
4258 %

```

The `\__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `\l__enumext_anspic_args_seq` sequence which will be processed by the `\__enumext_anspic_print:n` function in the second part of the definition of the `keyanspic` environment.

```

4259 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4260 {
4261     \__enumext_anspic_start_list_tag:
4262     \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4263     \__enumext_anspic_stop_start_list_tag:
4264     \bool_if:NTF \l__enumext_anspic_label_above_bool
4265     {
4266         \\[\__enumext_anspic_label_sep_skip] #3
4267     }
4268     {
4269         \\ #3
4270     }
4271     \__enumext_anspic_stop_list_tag:
4272 }

```

The value  $\langle n^{\circ} upper, n^{\circ} lower \rangle$  passed to the `layout-sty` key is split by comma and is handled directly by the function `\__enumext_anspic_print:n` and passed to the function `\__enumext_anspic_row:n`.

```

4273 \cs_new_protected:Nn \__enumext_anspic_print:n
4274 {
4275     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4276 }
4277 \cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }

```

The function `\__enumext_anspic_row:n` will set the *widths* for the `minipage` environments and place *all arguments* passed to `\anspic` saved in the `\l__enumext_anspic_args_seq` sequence inside them.

```

4278 \cs_new_protected:Nn \__enumext_anspic_row:n
4279 {
4280     \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4281     \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4282     \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4283     \int_step_inline:nnn
4284     { \l__enumext_anspic_above_int + 1 }
4285     { \l__enumext_anspic_below_int }
4286     {
4287         \IfDocumentMetadataTF
4288         {
4289             \tag_suspend:n {minipage}
4290         } { }
4291         \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4292             \centering
4293             \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4294         \end{minipage}
4295         \IfDocumentMetadataTF
4296         {
4297             \tag_resume:n {minipage}
4298         } { }
4299     }
4300     \par
4301 }

```

The `\__enumext_anspic_exec:` function will execute all the code in the `\anspic` command in the second argument of the `keyanspic` environment definition. If the key `layout-sty` is not set, everything will be printed on a *single line*.

```

4302 \cs_new_protected:Nn \__enumext_anspic_exec:
4303 {
4304     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4305     {
4306         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4307     }
4308     {
4309         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl

```

```

4310     }
4311 }

```

(End of definition for `\anspic` and others. This function is documented on page 16.)

### 13.43 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard  $\TeX$  list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makeLabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the `\item` command and the  $\langle content \rangle$  of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the  $\langle content \rangle$  passed to `\item`, we will also add the *optional argument* ( $\langle number \rangle$ ) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* ( $\langle number \rangle$ ).

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by  $\TeX$  and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

#### 13.43.1 Functions for item box width

We set the default value for the *width of the box* containing the  $\langle content \rangle$  of the items for `enumext*` environment.

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:
4312 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4313 {
4314   \dim_compare:nNt { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4315   {
4316     \dim_set:Nn \__enumext_columns_sep_vii_dim
4317     {
4318       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4319       / \__enumext_columns_vii_int
4320     }
4321   }
4322   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4323   \dim_set:Nn \__enumext_item_width_vii_dim
4324   {
4325     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4326     / \__enumext_columns_vii_int
4327     - \__enumext_labelwidth_vii_dim
4328     - \__enumext_labelsep_vii_dim
4329   }

```

When the key `rightmargin` is active we must adjust the values.

```

4330   \dim_compare:nNt { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4331   {
4332     \dim_sub:Nn \__enumext_item_width_vii_dim
4333     {
4334       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4335       / \__enumext_columns_vii_int
4336     }
4337     \dim_add:Nn \__enumext_columns_sep_vii_dim
4338     {
4339       \__enumext_rightmargin_vii_dim
4340     }
4341   }
4342 }

```

Same implementation for the `keyans*` environment.

```

4343 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4344 {
4345   \dim_compare:nNt { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4346   {
4347     \dim_set:Nn \__enumext_columns_sep_viii_dim

```

```

4348         {
4349             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4350             / \l__enumext_columns_viii_int
4351         }
4352     }
4353     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4354     \dim_set:Nn \l__enumext_item_width_viii_dim
4355     {
4356         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4357         / \l__enumext_columns_viii_int
4358         - \l__enumext_labelwidth_viii_dim
4359         - \l__enumext_labelsep_viii_dim
4360     }
4361     \dim_compare:nNt { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4362     {
4363         \dim_sub:Nn \l__enumext_item_width_viii_dim
4364         {
4365             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4366             / \l__enumext_columns_viii_int
4367         }
4368         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4369         {
4370             \l__enumext_rightmargin_viii_dim
4371         }
4372     }
4373 }

```

(End of definition for `\__enumext_starred_columns_set_vii:` and `\__enumext_starred_columns_set_viii:`)

### 13.43.2 Functions for join item columns

`\__enumext_starred_joined_item_vii:n`  
`\__enumext_starred_joined_item_viii:n`

The functions `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the *content* passed to `\item`(*columns*) will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4374 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4375 {
4376     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4377     \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4378     {
4379         \msg_warning:nnee { enumext } { item-joined }
4380         { \int_use:N \l__enumext_joined_item_vii_int }
4381         { \int_use:N \l__enumext_columns_vii_int }
4382         \int_set:Nn \l__enumext_joined_item_vii_int
4383         {
4384             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4385         }
4386     }
4387     \int_compare:nNt
4388     { \l__enumext_joined_item_vii_int }
4389     >
4390     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4391     {
4392         \msg_warning:nnee { enumext } { item-joined-columns }
4393         { \int_use:N \l__enumext_joined_item_vii_int }
4394         {
4395             \int_eval:n
4396             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4397         }
4398         \int_set:Nn \l__enumext_joined_item_vii_int
4399         {
4400             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4401         }
4402     }
4403     \int_compare:nNtTF { \l__enumext_joined_item_vii_int } > { 1 }
4404     {
4405         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4406         \int_decr:N \l__enumext_joined_item_aux_vii_int
4407         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4408         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4409         \dim_set:Nn \l__enumext_joined_width_vii_dim
4410         {
4411             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int

```

```

4412         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4413         + \l__enumext_columns_sep_vii_dim
4414         )*\l__enumext_joined_item_aux_vii_int
4415     }
4416     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4417 }
4418 {
4419     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4420     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4421 }
4422 }

```

Same implementation for the **keyans\*** environment.

```

4423 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4424 {
4425     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4426     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4427     {
4428         \msg_warning:nnee { enumext } { item-joined }
4429         { \int_use:N \l__enumext_joined_item_viii_int }
4430         { \int_use:N \l__enumext_columns_viii_int }
4431         \int_set:Nn \l__enumext_joined_item_viii_int
4432         {
4433             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4434         }
4435     }
4436     \int_compare:nNnT
4437     { \l__enumext_joined_item_viii_int }
4438     >
4439     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4440     {
4441         \msg_warning:nnee { enumext } { item-joined-columns }
4442         { \int_use:N \l__enumext_joined_item_viii_int }
4443         {
4444             \int_eval:n
4445             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4446         }
4447         \int_set:Nn \l__enumext_joined_item_viii_int
4448         {
4449             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4450         }
4451     }
4452     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4453     {
4454         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4455         \int_decr:N \l__enumext_joined_item_aux_viii_int
4456         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4457         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4458         \dim_set:Nn \l__enumext_joined_width_viii_dim
4459         {
4460             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4461             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4462             + \l__enumext_columns_sep_viii_dim
4463             )*\l__enumext_joined_item_aux_viii_int
4464         }
4465         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4466     }
4467     {
4468         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4469         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4470     }
4471 }

```

(End of definition for `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_viii:n`)

### 13.43.3 Functions for mini-env, mini-right and mini-right\* keys

`\__enumext_start_mini_vii:` The implementation of the `mini-env` key support is almost identical to the one used in the **enumext** and **keyans** environments, the difference is that the `\__enumext_mini_page` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

4472 \cs_new_protected:Nn \__enumext_start_mini_vii:

```

```

4473 {
4474   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4475   {
4476     \dim_set:Nn \l__enumext_minipage_left_vii_dim
4477     {
4478       \linewidth
4479       - \l__enumext_minipage_right_vii_dim
4480       - \l__enumext_minipage_hsep_vii_dim
4481     }
4482     \bool_set_true:N \l__enumext_minipage_active_vii_bool
4483     \dim_gset_eq:NN
4484       \g__enumext_minipage_right_vii_dim
4485       \l__enumext_minipage_right_vii_dim
4486     \__enumext_mini_addvspace_vii:
4487     \nointerlineskip\noindent
4488     \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4489   }
4490 }

```

The function `\__enumext_stop_mini_vii:` closes the `\__enumext_mini_page` environment on the “*left side*”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “*true*” which will be used in the function `\__enumext_after_env:n` to execute the `minipage` on the “*right side*”. At this point we will execute the `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `\__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.44).

```

4491 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4492 {
4493   \bool_if:NTF \l__enumext_minipage_active_vii_bool
4494   {
4495     \__enumext_stop_list:
4496     \__enumext_stop_store_level_vii:
4497     \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4498     \end__enumext_mini_page
4499     \hfill
4500     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4501   }
4502   {
4503     \__enumext_stop_list:
4504     \__enumext_stop_store_level_vii:
4505   }
4506 }

```

(End of definition for `\__enumext_start_mini_vii:` and `\__enumext_stop_mini_vii:`)

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `minipage` environment on the “*right side*”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

4507 \__enumext_after_env:n {enumext*}
4508 {
4509   \bool_if:NT \g__enumext_minipage_active_vii_bool
4510   {
4511     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4512     \legacy_if_gset_false:n { @minipage }
4513     \skip_vertical:N \c_zero_skip
4514     \par\addvspace { \g__enumext_minipage_right_skip }
4515     \bool_if:NF \g__enumext_minipage_center_vii_bool
4516     {
4517       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4518       {
4519         \centering
4520       }
4521     }
4522     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4523     {
4524       \tl_use:N \g__enumext_miniright_code_vii_tl
4525     }
4526     \box_use_drop:N \l__enumext_miniright_code_vii_box
4527     \skip_vertical:N \c_zero_skip
4528     \__enumext_endminipage:
4529     \par\addvspace{ \g__enumext_minipage_after_skip }

```

```

4530     }
4531     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4532     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4533     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4534     \dim_gzero:N \g__enumext_minipage_right_vii_dim
4535     \bool_gset_false:N \g__enumext_starred_bool
4536 }

```

`\__enumext_start_mini_viii:` The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumext*` environment.

`\__enumext_stop_mini_viii:`

```

4537 \cs_new_protected:Nn \__enumext_start_mini_viii:
4538 {
4539     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4540     {
4541         \dim_set:Nn \l__enumext_minipage_left_viii_dim
4542         {
4543             \linewidth
4544             - \l__enumext_minipage_right_viii_dim
4545             - \l__enumext_minipage_hsep_viii_dim
4546         }
4547         \bool_set_true:N \l__enumext_minipage_active_viii_bool
4548         \dim_gset_eq:NN
4549             \g__enumext_minipage_right_viii_dim
4550             \l__enumext_minipage_right_viii_dim
4551         \__enumext_mini_addvspace_viii:
4552         \nointerlineskip\noindent
4553         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4554     }
4555 }
4556 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4557 {
4558     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4559     {
4560         \__enumext_stop_list:
4561         \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4562         \end__enumext_mini_page
4563         \hfill
4564         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4565     }
4566     {
4567         \__enumext_stop_list:
4568     }
4569 }
4570 \__enumext_after_env:nn {keyans*}
4571 {
4572     \bool_if:NT \g__enumext_minipage_active_viii_bool
4573     {
4574         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4575         \par\addvspace { \g__enumext_minipage_right_skip }
4576         \bool_if:NF \g__enumext_minipage_center_viii_bool
4577         {
4578             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4579             {
4580                 \centering
4581             }
4582         }
4583         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4584         {
4585             \tl_use:N \g__enumext_miniright_code_viii_tl
4586         }
4587         \box_use_drop:N \l__enumext_miniright_code_viii_box
4588         \end__enumext_mini_page
4589         \par\addvspace{ \g__enumext_minipage_after_skip }
4590     }
4591     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4592     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4593     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4594     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4595 }

```

(End of definition for `\__enumext_start_mini_viii:` and `\__enumext_stop_mini_viii:`)

### 13.44 The environment enumext\*

`enumext*` First we will generate the environment and we will give a temporary definition to `\__enumext_stop_item_tmp_vii`: equal to `\__enumext_first_item_tmp_vii`: and next to `\item` equal to `\__enumext_start_item_tmp_vii`: which we will redefine later. Unlike the implementation used by the `shortlst` package, we will not set the values of `\rightskip` and `\@rightskip` equal to `\@flushglue` whose value is `0.0pt plus 1.0 fil`, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```

4596 \NewDocumentEnvironment{enumext*}{o}{
4597   {
4598     \__enumext_safe_exec_vii:
4599     \__enumext_parse_keys_vii:n {#1}
4600     \__enumext_before_list_vii:
4601     \__enumext_start_store_level_vii:
4602     \__enumext_start_list:nn { }
4603     {
4604       \__enumext_list_arg_two_vii:
4605       \__enumext_before_keys_exec_vii:
4606     }
4607     \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4608     \__enumext_starred_columns_set_vii:
4609     \item[] \scan_stop:
4610     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4611     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4612     \ignorespaces
4613   }
4614   {
4615     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4616     \__enumext_stop_item_tmp_vii:
4617     \__enumext_remove_extra_parsep_vii:
4618     \__enumext_after_list_vii:
4619   }

```

(End of definition for `enumext*`. This function is documented on page 5.)

`\__enumext_safe_exec_vii:` We will first call the function `\__enumext_is_not_nested`: which sets `\g__enumext_starred_bool` to true if we are NOT nested within `enumext`, then call the function `\__enumext_internal_mini_page`: to create the environment `__enumext_mini_page`, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `\__enumext_is_on_first_level`: which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the “storage system” to be used.

```

4620 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4621   {
4622     \__enumext_is_not_nested:
4623     \__enumext_internal_mini_page:
4624     \int_incr:N \l__enumext_level_h_int
4625     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4626     {
4627       \msg_error:nn { enumext } { nested }
4628     }
4629     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4630     {
4631       \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4632     }
4633     \bool_set_true:N \l__enumext_starred_bool
4634     \bool_set_false:N \l__enumext_standar_bool
4635     \__enumext_is_on_first_level:
4636   }

```

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

`\__enumext_parse_keys_vii:n` First we will clear the variable `\l__enumext_series_str` used by the key `series`, process the environment `[⟨key = val⟩]` and execute the function `\__enumext_parse_series:n` and used by the key `series`, then we execute the function `\__enumext_store_active_keys_vii:n` and reprocess the `(keys)` to pass them to the storage `sequence` if the key `save-key` is not active.

```

4637 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4638   {
4639     \tl_if_novalue:nF {#1}
4640     {
4641       \str_clear:N \l__enumext_series_str

```



```

4642         \keys_set:nn { enumext / enumext* } {#1}
4643         \__enumext_parse_series:n {#1}
4644         \__enumext_store_active_keys_vii:n {#1}
4645     }
4646 }

```

(End of definition for \\_\_enumext\_parse\_keys\_vii:n.)

\\_\_enumext\_before\_list\_vii: The function \\_\_enumext\_before\_list\_vii: first calls the function \\_\_enumext\_vspace\_above\_vii: used by the keys `above` and `above*`, then calls the function \\_\_enumext\_check\_ans\_active: for the check answer mechanism and finally calls the functions \\_\_enumext\_before\_args\_exec: and \\_\_enumext\_start\_mini\_vii: used by the keys `before*`, `mini-env`, `mini-right` and `mini-right*`.

```

4647 \cs_new_protected:Nn \__enumext_before_list_vii:
4648 {
4649     \__enumext_vspace_above_vii:
4650     \__enumext_check_ans_active:
4651     \__enumext_before_args_exec_vii:
4652     \__enumext_start_mini_vii:
4653 }

```

(End of definition for \\_\_enumext\_before\_list\_vii:.)

\\_\_enumext\_after\_list\_vii: The function \\_\_enumext\_after\_list\_vii: first calls the function \\_\_enumext\_stop\_mini\_vii: which internally calls \\_\_enumext\_stop\_list: and \\_\_enumext\_stop\_store\_level\_vii: (§13.43.3) used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions \\_\_enumext\_after\_stop\_list\_vii: used by the key `after`, \\_\_enumext\_check\_ans\_key\_hook: used by the key `check-ans`, \\_\_enumext\_vspace\_below\_vii: used by the keys `below` and `below*`. Finally set \l\_\_enumext\_starred\_bool to false and call the \\_\_enumext\_resume\_save\_counter: function used by the `series`, `resume` and `resume*` keys.

```

4654 \cs_new_protected:Nn \__enumext_after_list_vii:
4655 {
4656     \__enumext_stop_mini_vii:
4657     \__enumext_after_stop_list_vii:
4658     \__enumext_check_ans_key_hook:
4659     \__enumext_vspace_below_vii:
4660     \bool_set_false:N \l__enumext_starred_bool
4661     \__enumext_resume_save_counter:
4662 }

```

(End of definition for \\_\_enumext\_after\_list\_vii:.)

\\_\_enumext\_start\_store\_level\_vii: The \\_\_enumext\_start\_store\_level\_vii: and \\_\_enumext\_stop\_store\_level\_vii: functions activate the “storing structure” mechanism in *sequence* for \anskey command and `anskey*` environment if `enumext*` are nested in `enumext`.

```

4663 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4664 {
4665     \bool_if:NT \l__enumext_store_active_bool
4666     {
4667         \int_compare:nNt { \l__enumext_level_int } > { 0 }
4668         {
4669             \__enumext_store_level_open_vii:
4670         }
4671     }
4672 }
4673 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4674 {
4675     \bool_if:NT \l__enumext_store_active_bool
4676     {
4677         \int_compare:nNt { \l__enumext_level_int } > { 0 }
4678         {
4679             \__enumext_store_level_close_vii:
4680         }
4681     }
4682 }

```

(End of definition for \\_\_enumext\_start\_store\_level\_vii: and \\_\_enumext\_stop\_store\_level\_vii:.)

### 13.44.1 The command `\item` in `enumext*`

`\__enumext_first_item_tmp_vii:`

The `\__enumext_first_item_tmp_vii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the “*first*” `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_vii:` function inside the environment body definition.

```
4683 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4684 {
4685     \skip_horizontal:n
4686     {
4687         -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim
4688     }
4689     \ignorespaces
4690 }
```

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

`\__enumext_start_item_tmp_vii:`

`\__enumext_item_peek_args_vii:`

`\__enumext_joined_item_vii:w`

`\__enumext_standar_item_vii:w`

`\__enumext_starred_item_vii:w`

First we will call the function `\__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\__enumext_item_column_pos_vii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item’s in the environment. After that we will call the function `\__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```
4691 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4692 {
4693     \__enumext_stop_item_tmp_vii:
4694     \int_incr:N \__enumext_item_column_pos_vii_int
4695     \int_gincr:N \g__enumext_item_count_all_vii_int
4696     \__enumext_item_peek_args_vii:
4697 }
```

The function `\__enumext_item_peek_args_vii:` will handle the `\item(<number>)`. Look for the argument “(”, if it is present we will call the function `\__enumext_joined_item_vii:w (<number>)`, which is in charge of joining the item’s in the same row, in case they are not present we will set the default value (1).

```
4698 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4699 {
4700     \peek_meaning:NTF (
4701         { \__enumext_joined_item_vii:w }
4702         { \__enumext_joined_item_vii:w (1) }
4703     }
```

The function `\__enumext_joined_item_vii:w` will first call the function `\__enumext_starred_joined_item_vii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “\*”, if it is present we will call the function `\__enumext_starred_item_vii:w` otherwise we will call the function `\__enumext_standar_item_vii:w`.

```
4704 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4705 {
4706     \__enumext_starred_joined_item_vii:n {#1}
4707     \peek_meaning_remove:NTF *
4708     { \__enumext_starred_item_vii:w }
4709     { \__enumext_standar_item_vii:w }
4710 }
```

The function `\__enumext_standar_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `\__enumext_wrap_label_opt_vii_bool` equal to the state of the variable `\__enumext_wrap_label_opt_vii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[<custom>]` by means of the function `\__enumext_start_item_vii:w`, otherwise we will set the value of the variable `\__enumext_wrap_label_vii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]`.

```
4711 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4712 {
4713     \bool_set_false:N \__enumext_item_starred_vii_bool
4714     \peek_meaning:NTF [
4715     {
4716         \bool_set_eq:NN \__enumext_wrap_label_vii_bool \__enumext_wrap_label_opt_vii_bool
4717         \__enumext_start_item_vii:w
4718     }
4719     {
4720         \bool_set_true:N \__enumext_wrap_label_vii_bool
4721         \legacy_if_set_true:n { @noitemarg }
4722         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
4723     }
4724 }
```

The function `\__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]`.

```

4725 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4726 {
4727   \bool_set_true:N \__enumext_item_starred_vii_bool
4728   \bool_set_true:N \__enumext_wrap_label_vii_bool
4729   \peek_meaning:NTF [
4730     { \__enumext_starred_item_vii_aux_i:w }
4731     { \__enumext_starred_item_vii_aux_ii:w }
4732   }
4733 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4734 {
4735   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4736   \__enumext_starred_item_vii_aux_ii:w
4737 }
4738 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4739 {
4740   \peek_meaning:NTF [
4741     { \__enumext_starred_item_vii_aux_iii:w }
4742     {
4743       \dim_set_eq:NN \__enumext_item_symbol_sep_vii_dim \__enumext_labelsep_vii_dim
4744       \legacy_if_set_true:n { @noitemarg }
4745       \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
4746     }
4747   }
4748 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4749 {
4750   \dim_set:Nn \__enumext_item_symbol_sep_vii_dim {#1}
4751   \legacy_if_set_true:n { @noitemarg }
4752   \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
4753 }

```

(End of definition for `\__enumext_start_item_tmp_vii:` and others.)

`\__enumext_fake_make_label_vii:n`

The `\__enumext_fake_make_label_vii:n` function will be in charge of handling our definition of `\item`. First we increment the counter `enumXvii` for the enumerated items and activate support for the *check answers* mechanism, followed by support for `\item*[\langle symbol \rangle][\langle offset \rangle]` if present, then the `wrap-label` and `wrap-label*` keys which we execute using `\makebox` whose width will be given by the `labelwidth` key and position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute the `labelsep` key applying a `\skip_horizontal:N` and `\ignorespaces`.

For compatibility with *tagged* PDF and *hyperref* when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to “true”. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier. This patch is only needed if you are running `pdflatex` and not if you are running `lualatex`

```

4754 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4755 {
4756   \legacy_if:nT { @noitemarg }
4757   {
4758     \legacy_if_set_false:n { @noitemarg }
4759     \legacy_if:nT { @nmbrrlist }
4760     {
4761       \IfDocumentMetadataTF
4762       {
4763         \bool_if:NT \__enumext_hyperref_bool
4764         {
4765           \legacy_if_set_true:n { @hyper@item }
4766         }
4767       } { }
4768       \refstepcounter{enumXvii}
4769       \bool_if:NT \__enumext_check_answers_bool
4770       {
4771         \int_gincr:N \g__enumext_item_number_int
4772         \bool_set_true:N \__enumext_item_number_bool
4773       }
4774     }
4775   }
4776   \bool_if:NT \__enumext_item_starred_vii_bool
4777   {
4778     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl

```

```

4779         {
4780             \tl_gset_eq:NN
4781             \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4782         }
4783         \mode_leave_vertical:
4784         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4785         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4786         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4787         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4788     }
4789     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4790     {
4791         \tl_use:N \l__enumext_label_font_style_vii_tl
4792         \bool_if:NTF \l__enumext_wrap_label_vii_bool
4793         {
4794             \__enumext_wrapper_label_vii:n {#1}
4795         }
4796         { #1 }
4797     }
4798     \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4799 }

```

(End of definition for `\__enumext_fake_make_label_vii:n`.)

### 13.44.2 Real definition of `\item` in `enumext*`

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`\__enumext_start_item_vii:w` The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to `\__enumext_stop_item_vii:` which we will define later, after that we will start capturing `\item` and “*item content*” in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```

4800 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4801 {
4802     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4803     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4804     {
4805         \l__enumext_joined_width_vii_dim
4806         + \l__enumext_labelwidth_vii_dim
4807         + \l__enumext_labelsep_vii_dim
4808     }

```

Redefine the `\footnote` command.

```
4809 \__enumext_renew_footnote_starred:
```

Now we insert our *sockets* for *tagging* PDF support and run `\item`.

```

4810 \__enumext_start_list_tag:n {enumext*}
4811 \__enumext_fake_make_label_vii:n {#1}
4812 \__enumext_stop_start_list_tag:

```

Finally we open the `minipage` environment, capture the “*item content*”, make `\parindent` take the value of the key `listparindent` and `\parskip` take the value of the key `parsep`, then execute the keys `itemindent` and `first`.

- Here the use of `\unskip` and `\skip_horizontal:n` with the value of `listparindent` is necessary, otherwise an unwanted space is created when using `\item[⟨opt⟩]` and the value passed to the key `itemindent` is incremented.

```

4813 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4814 \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4815 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4816 \__enumext_unskip_unkern:
4817 \__enumext_unskip_unkern:
4818 \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4819 \tl_use:N \l__enumext_fake_item_indent_vii_tl
4820 \tl_use:N \l__enumext_after_list_args_vii_tl
4821 }

```

The `\__enumext_stop_item_vii:` function will finish the fetching `\item` and “*item content*” by closing the `minipage` environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```

4822 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4823 {
4824     \__enumext_endminipage:
4825     \__enumext_stop_list_tag:n {enumext*}
4826     \hbox_set_end:

```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print `\item` and “*item content*” from the *horizontal box*.,

```
4827 \int_set:Nn \hbadness { 10000 }
4828 \box_use_drop:N \l__enumext_item_text_vii_box
```

Finally apply the *vertical space* between rows set by `itemsep` key passed to `\parsep` using `\par\noindent` and *horizontal space* between columns set by `columns-sep` key using `\skip_horizontal:N`.

```
4829 \int_compare:nNnTF
4830 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4831 {
4832   \par\noindent
4833   \int_zero:N \l__enumext_item_column_pos_vii_int
4834 }
4835 {
4836   \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4837 }
4838 }
```

(End of definition for `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:.`)

`\__enumext_remove_extra_parsep_vii:`

Remove the extra *vertical space* equal to `\parsep=itemsep` when the total number of `\item` is divisible by the number of `\item` in the last row of the environment. Here the use of `\unskip` or `\removeatlastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in *(vertical mode)*.

```
4839 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4840 {
4841   \int_compare:nNnT
4842   {
4843     \int_mod:nn
4844     { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4845   }
4846   =
4847   { 0 }
4848   {
4849     \para_end:
4850     \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4851     \skip_vertical:N \c_zero_skip
4852     \int_gzero:N \g__enumext_item_count_all_vii_int
4853   }
4854 }
```

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

As we don’t want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the “hook” function `\__enumext_after_env:nn`.

```
4855 \__enumext_after_env:nn {enumext*}
4856 {
4857   \__enumext_execute_after_env:
4858 }
```

### 13.45 The environment `keyans*`

`keyans*`

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `\__enumext_check_starred_cmd:n` function added in the second part.

```
4859 \NewDocumentEnvironment{keyans*}{ o }
4860 {
4861   \__enumext_safe_exec_viii:
4862   \__enumext_parse_keys_viii:n {#1}
4863   \__enumext_before_list_viii:
4864   \__enumext_start_list:nn { }
4865   {
4866     \__enumext_list_arg_two_viii:
4867     \__enumext_before_keys_exec_viii:
4868   }
4869   \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4870   \__enumext_starred_columns_set_viii:
4871   \item[] \scan_stop:
4872   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4873   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4874   \ignorespaces
4875 }
```

```

4876 {
4877   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4878   \__enumext_stop_item_tmp_viii:
4879   \__enumext_remove_extra_parsep_viii:
4880   \__enumext_check_starred_cmd:n { item }
4881   \__enumext_after_list_viii:
4882 }

```

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

`\__enumext_safe_exec_viii:` The `\__enumext_safe_exec_viii:` function will first check if the `save-ans` key is active and only when this is true the environment will be available, it will increment the value of `\l__enumext_keyans_level_h_int` and return an error message when we are nesting the environment, then it will call the `\__enumext_keyans_name_and_start:` function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest `keyans*` in `enumext*` returning an error and we will set `\l__enumext_starred_bool` to true, finally we will check if we are within the appropriate level within the `enumext` environment.

```

4883 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4884 {
4885   \bool_if:NF \l__enumext_store_active_bool
4886   {
4887     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4888   }
4889   \int_incr:N \l__enumext_keyans_level_h_int
4890   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4891   {
4892     \msg_error:nn { enumext } { nested }
4893   }
4894   \__enumext_keyans_name_and_start:
4895   \bool_if:NT \l__enumext_starred_bool
4896   {
4897     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4898   }
4899   \bool_set_true:N \l__enumext_starred_bool
4900   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4901   \bool_set_false:N \l__enumext_store_active_bool
4902   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4903   {
4904     \msg_error:nn { enumext } { keyans-wrong-level }
4905   }
4906 }

```

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

`\__enumext_parse_keys_viii:n` Parse [`<key = val>`] for `keyans*`.

```

4907 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4908 {
4909   \tl_if_novalue:nF {#1}
4910   {
4911     \keys_set:nn { enumext / keyans* } {#1}
4912   }
4913 }

```

(End of definition for `\__enumext_parse_keys_viii:n`.)

`\__enumext_before_list_viii:` The function `\__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{<code>}` defined by the `before*` key if it is active, the call the function `\__enumext_start_mini_viii:` handle by `mini-env`.

```

4914 \cs_new_protected:Nn \__enumext_before_list_viii:
4915 {
4916   \__enumext_vspace_above_viii:
4917   \__enumext_before_args_exec_viii:
4918   \__enumext_start_mini_viii:
4919 }

```

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

`\__enumext_after_list_viii:` The function `\__enumext_after_list_viii:` first call the function `\__enumext_stop_mini_viii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

4920 \cs_new_protected:Nn \__enumext_after_list_viii:
4921 {
4922     \__enumext_stop_mini_viii:
4923     \__enumext_after_stop_list_viii:
4924     \__enumext_vspace_below_viii:
4925 }

```

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

### 13.45.1 The command `\item` in `keyans*`

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the *optional argument* (`\langle number \rangle`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\langle label \rangle` next to the `[\langle content \rangle]` if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*[\langle content \rangle]`, `\item(\langle number \rangle)*` and `\item(\langle number \rangle)*[\langle content \rangle]` commands.

`\__enumext_first_item_tmp_viii:` The `\__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the “*first*” `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```

4926 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4927 {
4928     \skip_horizontal:n
4929     {
4930         -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
4931     }
4932     \ignorespaces
4933 }

```

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

`\__enumext_start_item_tmp_viii:` First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. `\__enumext_item_peek_args_viii:` After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

4934 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4935 {
4936     \__enumext_stop_item_tmp_viii:
4937     \int_incr:N \l__enumext_item_column_pos_viii_int
4938     \int_gincr:N \g__enumext_item_count_all_viii_int
4939     \__enumext_item_peek_args_viii:
4940 }

```

The function `\__enumext_item_peek_args_viii:` will handle the `\item(\langle number \rangle)`. Look for the argument “`(`”, if it is present we will call the function `\__enumext_joined_item_viii:w` (`\langle number \rangle`), which is in charge of joining the item’s in the same row, in case they are not present we will set the default value `(1)`.

```

4941 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4942 {
4943     \peek_meaning:NTF (
4944         { \__enumext_joined_item_viii:w }
4945         { \__enumext_joined_item_viii:w (1) }
4946     }

```

The function `\__enumext_joined_item_viii:w` will first call the function `\__enumext_starred_joined_item_viii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “`*`”, if it is present we will call the function `\__enumext_starred_item_viii:w` otherwise we will call the function `\__enumext_standar_item_viii:w`.

```

4947 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4948 {
4949     \__enumext_starred_joined_item_viii:n {#1}
4950     \peek_meaning_remove:NTF *
4951     { \__enumext_starred_item_viii:w }
4952     { \__enumext_standar_item_viii:w }
4953 }

```



The function `\__enumext_standar_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[⟨custom⟩]` by means of the function `\__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]`.

```

4954 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4955 {
4956   \bool_set_false:N \l__enumext_item_starred_viii_bool
4957   \peek_meaning:NTF [
4958     {
4959       \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4960       \__enumext_start_item_viii:w
4961     }
4962     {
4963       \bool_set_true:N \l__enumext_wrap_label_viii_bool
4964       \legacy_if_set_true:n { @noitemarg }
4965       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4966     }
4967   }

```

(End of definition for `\__enumext_start_item_tmp_viii:` and others.)

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
\__enumext_starred_item_exec:

```

The function `\__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[⟨content⟩]`.

```

4968 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4969 {
4970   \bool_set_true:N \l__enumext_item_starred_viii_bool
4971   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4972   \peek_meaning:NTF [
4973     { \__enumext_starred_item_viii_aux_i:w }
4974     { \__enumext_starred_item_viii_aux_ii:w }
4975   }

```

The function `\__enumext_starred_item_viii_aux_i:w` will save the *optional argument* to `\item*` in `\l__enumext_store_current_opt_arg_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `\l__enumext_store_current_label_tl` if present, then call the function `\__enumext_starred_item_viii_aux_ii:w`.

```

4976 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4977 {
4978   \tl_clear:N \l__enumext_store_current_label_tl
4979   \tl_if_no_value:nF { #1 }
4980   {
4981     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4982     {
4983       \tl_put_right:Ne \l__enumext_store_current_label_tl
4984       {
4985         \l__enumext_store_keyans_item_opt_sep_tl
4986       }
4987       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4988     }
4989     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4990   }
4991   \__enumext_starred_item_viii_aux_ii:w
4992 }
4993 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4994 {
4995   \legacy_if_set_true:n { @noitemarg }
4996   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4997 }

```

The function `\__enumext_starred_item_exec:` will be in charge of storing the current *⟨label⟩* for `\item*` followed by the *[⟨content⟩]* for `\item*[⟨content⟩]` if present in the *sequence* and *prop list* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

4998 \cs_new_protected:Nn \__enumext_starred_item_exec:
4999 {
5000   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5001   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5002   \__enumext_keyans_store_ref:
5003   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }

```

```

5004 \__enumext_keyans_addto_seq_link:
5005 \int_gincr:N \g__enumext_check_starred_cmd_int
5006 \bool_if:NT \l__enumext_show_answer_bool
5007 {
5008   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
5009 }
5010 \bool_if:NT \l__enumext_show_position_bool
5011 {
5012   \tl_set:Nx \l__enumext_mark_answer_sym_tl
5013   {
5014     \group_begin:
5015     \exp_not:N \normalfont
5016     \exp_not:N \footnotesize [ \int_eval:n
5017       {
5018         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5019       }
5020     ]
5021     \group_end:
5022   }
5023   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
5024 }
5025 }

```

(End of definition for \\_\_enumext\_starred\_item\_viii:w and others.)

\\_\_enumext\_fake\_make\_label\_viii:n

The implementation at this is very similar to that of the `enumext*` environment.

```

5026 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5027 {
5028   \legacy_if:nT { @noitemarg }
5029   {
5030     \legacy_if_set_false:n { @noitemarg }
5031     \legacy_if:nT { @nmbrrlist }
5032     {
5033       \refstepcounter{enumXviii}
5034     }
5035   }
5036   \bool_if:NT \l__enumext_item_starred_viii_bool
5037   {
5038     \__enumext_starred_item_exec:
5039   }
5040   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5041   {
5042     \tl_use:N \l__enumext_label_font_style_viii_tl
5043     \bool_if:NTF \l__enumext_wrap_label_viii_bool
5044     {
5045       \__enumext_wrapper_label_viii:n {#1}
5046     }
5047     { #1 }
5048   }
5049   \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5050 }

```

(End of definition for \\_\_enumext\_fake\_make\_label\_viii:n.)

### 13.45.2 Real definition of \item in keyans\*

\\_\_enumext\_start\_item\_viii:w

The implementation at this is very similar to that of the `enumext*` environment.

\\_\_enumext\_stop\_item\_viii:

```

5051 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5052 {
5053   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5054   \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5055   {
5056     \l__enumext_joined_width_viii_dim
5057     + \l__enumext_labelwidth_viii_dim
5058     + \l__enumext_labelsep_viii_dim
5059   }
5060   \__enumext_renew_footnote_starred:
5061   \__enumext_start_list_tag:n {keyans*}
5062   \__enumext_fake_make_label_viii:n {#1}
5063   \__enumext_stop_start_list_tag:
5064   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
5065   \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim

```

```

5066 \skip_set_eq:Nn \parskip \l__enumext_parsep_viii_skip
5067 \__enumext_unskip_unkern:
5068 \__enumext_unskip_unkern:
5069 \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5070 \tl_use:N \l__enumext_fake_item_indent_viii_tl
5071 \bool_if:NT \l__enumext_item_starred_viii_bool
5072 {
5073   \__enumext_keyans_show_item_opt:
5074 }
5075 \tl_use:N \l__enumext_after_list_args_viii_tl
5076 }
5077 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5078 {
5079   \__enumext_endminipage:
5080   \__enumext_stop_list_tag:n {keyans*}
5081   \hbox_set_end:
5082   \int_set:Nn \hbadness { 10000 }
5083   \box_use_drop:N \l__enumext_item_text_viii_box
5084   \int_compare:nNnTF
5085     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5086     {
5087       \par\noindent
5088       \int_zero:N \l__enumext_item_column_pos_viii_int
5089     }
5090     {
5091       \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5092     }
5093 }

```

(End of definition for \\_\_enumext\_start\_item\_viii:w and \\_\_enumext\_stop\_item\_viii:.)

\\_\_enumext\_remove\_extra\_parsep\_viii:

The implementation at this is very similar to that of the `enumext*` environment.

```

5094 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5095 {
5096   \int_compare:nNnT
5097     {
5098       \int_mod:nn
5099         { \g__enumext_item_count_all_viii_int }
5100         { \l__enumext_columns_viii_int }
5101     }
5102     =
5103     { 0 }
5104     {
5105       \para_end:
5106       \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5107       \skip_vertical:N \c_zero_skip
5108       \int_gzero:N \g__enumext_item_count_all_viii_int
5109     }
5110 }

```

(End of definition for \\_\_enumext\_remove\_extra\_parsep\_viii:.)

### 13.46 The command \getkeyans

\getkeyans  
 \\_\_enumext\_getkeyans\_aux:n  
 \\_\_enumext\_getkeyans:n

The `\getkeyans` command takes a *mandatory argument* of the form  $\langle \textit{store name} : \textit{position} \rangle$ . Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-anskey`.

```

5111 \NewDocumentCommand \getkeyans { m }
5112 {
5113   \exp_args:Ne \__enumext_getkeyans_aux:n
5114     { \tl_to_str:e { \text_expand:n {#1} } }
5115 }

```

The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “.”. If “.” is omitted it will return an error.

```

5116 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5117 {
5118   \str_if_in:nnTF {#1} { : }
5119   {
5120     \use:e
5121     {
5122       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:

```

```

5123         { {##1} {##2} }
5124     }
5125     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5126 }
5127 { \msg_error:nnn { enumext } { missing-colon } {#1} }
5128 }

```

The internal function `\__enumext_getkeyans:nn` will check for the existence of the *prop list*, if it does not exist it will return an error message, then it will fetch the content specified by the *second argument* from *prop list*.

```

5129 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5130 {
5131     \prop_if_exist:cTF { g__enumext_#1_prop }
5132     {
5133         \prop_item:cn { g__enumext_#1_prop }{#2}
5134     }
5135     {
5136         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5137     }
5138 }

```

(End of definition for `\getkeyans`, `\__enumext_getkeyans_aux:n`, and `\__enumext_getkeyans:nn`. This function is documented on page 17.)

### 13.47 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the *sequence* defined by the `save-ans` key. The first thing we will do is define a set of *filtered keys* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default *keys* for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[⟨print,*⟩]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[⟨print,level⟩]`.

```

5139 \keys_define:nn { enumext / print }
5140 {
5141     print* .code:n      = \keys_precompile:neN { enumext / enumext* }
5142                     { \__enumext_filter_save_key:n {#1} }
5143                     \l__enumext_print_keyans_starred_tl, % starred cmd
5144     print* .initial:n   = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5145     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5146                     { \__enumext_filter_save_key:n {#1} }
5147                     \l__enumext_print_keyans_i_tl,
5148     print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5149     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5150                     { \__enumext_filter_save_key:n {#1} }
5151                     \l__enumext_print_keyans_ii_tl,
5152     print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
5153     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5154                     { \__enumext_filter_save_key:n {#1} }
5155                     \l__enumext_print_keyans_iii_tl,
5156     print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
5157     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5158                     { \__enumext_filter_save_key:n {#1} }
5159                     \l__enumext_print_keyans_iv_tl,
5160     print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
5161     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5162                     { \__enumext_filter_save_key:n {#1} }
5163                     \l__enumext_print_keyans_vii_tl, % starred nested
5164     print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
5165 }

```

- The reason for storing *keys* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

`\printkeyans`

`\__enumext_printkeyans:nnn`

Create a user command to print “all stored content” in *sequence* for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `\__enumext_printkeyans:nnn`.

```

5166 \NewDocumentCommand \printkeyans { s O{} m }

```

```

5167 {
5168   \group_begin:
5169     \tl_use:N \l__enumext_print_keyans_i_tl
5170     \tl_use:N \l__enumext_print_keyans_ii_tl
5171     \tl_use:N \l__enumext_print_keyans_iii_tl
5172     \tl_use:N \l__enumext_print_keyans_iv_tl
5173     \tl_use:N \l__enumext_print_keyans_vii_tl
5174     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5175   \group_end:
5176 }

```

The internal function `\__enumext_printkeyans:nnn` will check for the existence of the *sequence*, if it does not exist it will return an error message, then it will check if not empty.

```

5177 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5178 {
5179   \seq_if_exist:cTF { g__enumext_#3_seq }
5180   {
5181     \seq_if_empty:cF { g__enumext_#3_seq }
5182     {

```

If the *starred argument* ‘\*’ is present we will check that the environment `enumext*` is not saved in the *sequence*, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default *⟨keys⟩* for the environment `enumext*`, we set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument* and map the *sequence*, then set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```

5183     \bool_if:nTF {#1}
5184     {
5185       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5186       {
5187         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5188       }
5189       {
5190         \tl_use:N \l__enumext_print_keyans_starred_tl
5191         \bool_set_true:N \l__enumext_base_line_fix_bool
5192         \bool_set_true:N \l__enumext_print_keyans_star_bool
5193         \begin{enumext*}[#2]
5194           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5195           \end{enumext*}
5196           \bool_set_false:N \l__enumext_base_line_fix_bool
5197           \bool_set_false:N \l__enumext_print_keyans_star_bool
5198         }
5199       }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “first level” then map the *sequence*.

```

5200       {
5201         \begin{enumext}[#2]
5202         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5203         \end{enumext}
5204       }
5205     }
5206   }
5207   {
5208     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5209   }
5210 }

```

(End of definition for `\printkeyans` and `\__enumext_printkeyans:nnn`. This function is documented on page 18.)

### 13.48 The command `\setenumext`

The command `\setenumext` will be in charge of managing the *⟨keys⟩* passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` environment and “first level” of the `enumext` environment so as not to capture *⟨keys⟩* that complicate us.

The function `\__enumext_filter_first_level:n` will be in charge of filtering the *⟨keys⟩* passed to the environment `enumext*` and “first level” of the environment `enumext`.

```

\__enumext_filter_first_level:n
\__enumext_filter_first_level_key:n
\__enumext_filter_first_level_pair:nn
5211 \cs_new:Npn \__enumext_filter_first_level:n #1
5212 {
5213   \use:e
5214   {

```

```

5215         \keyval_parse:NNn
5216         \__enumext_filter_first_level_key:n
5217         \__enumext_filter_first_level_pair:nn {#1}
5218     }
5219 }

```

The function `\__enumext_filter_first_level_key:n` will be responsible for filtering the *⟨keys⟩* that are passed “without value” by excluding the keys `resume` and `resume*`.

```

5220 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5221 {
5222     \str_case:nnF {#1}
5223     {
5224         { resume } {}
5225         { resume* } {}
5226     }
5227     { , { \exp_not:n {#1} } }
5228 }

```

The function `\__enumext_filter_first_level_pair:nn` will be responsible for filtering the *⟨keys⟩* that are passed “with value” by excluding the `series`, `resume` and `save-ans` keys.

```

5229 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5230 {
5231     \str_case:nnF {#1}
5232     {
5233         { series } {}
5234         { resume } {}
5235         { save-ans } {}
5236     }
5237     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
5238 }

```

(End of definition for `\__enumext_filter_first_level:n`, `\__enumext_filter_first_level_key:n`, and `\__enumext_filter_first_level_pair:nn`.)

Now define a “meta families” of *⟨keys⟩* to access from `\setenumext`.

```

5239 \keys_define:nn { enumext / meta-families }
5240 {
5241     enumext-1 .code:n =
5242     {
5243         \keys_set:ne { enumext / level-1 }
5244         {
5245             \__enumext_filter_first_level:n {#1}
5246         }
5247     } ,
5248     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5249     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5250     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5251     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5252     enumext* .code:n =
5253     {
5254         \keys_set:ne { enumext / enumext* }
5255         {
5256             \__enumext_filter_first_level:n {#1}
5257         }
5258     } ,
5259     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5260     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5261     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5262     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5263     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5264     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5265     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5266     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5267 }

```

We store them in the constant sequence `\c__enumext_all_families_seq` separated by commas.

```

5268 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5269 {
5270     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5271     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5272 }

```

```

\setenumext
__enumext_set_parse:n
__enumext_set_error:nn
5273 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5274 {
5275   \seq_clear:N \__enumext_setkey_tmpa_seq
5276   \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5277   \int_set:Nn \__enumext_setkey_tmpa_int
5278   {
5279     \seq_count:N \__enumext_setkey_tmpb_seq
5280   }
5281   \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5282   {
5283     \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
5284     \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5285     \seq_set_map_e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
5286     {
5287       \tl_use:N \__enumext_setkey_tmpa_tl - ##1
5288     }
5289   }
5290   {
5291     \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5292   }
5293   \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
5294   { \seq_map_inline:Nn \c__enumext_all_families_seq }
5295   { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
5296   {
5297     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5298   }
5299 }

```

Internal functions used by the `\setenumext` command.

```

5300 \cs_new_protected:Npn \__enumext_set_parse:n #1
5301 {
5302   \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5303   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5304   { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
5305   \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
5306   {
5307     \seq_put_right:Ne \__enumext_setkey_tmpa_seq
5308     { \tl_trim_spaces:n {#1} }
5309   }
5310   { \__enumext_set_error:nn {#1} { } }
5311 }
5312 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5313 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\setenumext`, `\__enumext_set_parse:n`, and `\__enumext_set_error:nn`. This function is documented on page 6.)

### 13.49 The command `\setenumextmeta`

The command `\setenumextmeta` will be responsible for adding new “meta-keys” for the `enumext` and `enumext*` environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in [Add .meta key to existing keys \(l3keys\)](#).

```

\setenumextmeta
\c__enumext_meta_paths_prop
__enumext_add_meta_key:nnn
__enumext_def_meta_key:nnn
__enumext_def_meta_key:Vnn
5314 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5315 {
5316   {enumext,1} = level-1,
5317   {enumext,2} = level-2,
5318   {enumext,3} = level-3,
5319   {enumext,4} = level-4,
5320   {enumext*} = enumext*
5321 }

```

Now we create the user command taking care that unknown cannot be passed as an argument.

```

5322 \NewDocumentCommand \setenumextmeta { s 0{enumext,1} m +m }
5323 {
5324   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5325   { \msg_error:nn { enumext } { prohibited-unknown } }
5326   {
5327     \bool_if:nTF {#1}
5328     {
5329       \int_step_inline:nn { 4 }

```



```

5330         { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5331         \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5332     }
5333     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5334 }
5335 }

```

The internal functions `\__enumext_add_meta_key:nnn` and `\__enumext_def_meta_key:nnn` will check the *optional argument* and create the “*meta-key*”.

```

5336 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5337 {
5338     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5339     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5340     \prop_get:NVNTF
5341     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5342     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5343     {
5344         \msg_error:nnn { enumext } { unknown-set } {#1}
5345         \use_none:nn
5346     }
5347 }
5348 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5349 {
5350     \bool_lazy_or:nnTF
5351     { \keys_if_exist:p:nn { enumext / #1 } {#2} }
5352     { \keys_if_exist:p:nn { enumext / enumext* } {#2} }
5353     { \msg_error:nnn { enumext } { already-defined } {#2} }
5354     {
5355         \keys_define:nn { enumext / #1 }
5356         {
5357             #2 .meta:n = {#3},
5358             #2 .value_forbidden:n = true
5359         }
5360     }
5361 }
5362 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

(End of definition for `\setenumextmeta` and others. This function is documented on page 6.)

### 13.50 The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in [Expand a .cs defined by key inside the function](#).

`\foreachkeyans`

```

\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n
\__enumext_foreach_keyans:nn
\__enumext_foreach_add_body:n

```

We define a set of *⟨keys⟩* for command and we will save the default values of these in `\g__enumext_foreach_default_keys_tl` to avoid the use of group.

```

5363 \keys_define:nn { enumext / foreach }
5364 {
5365     before .tl_set:N = \l__enumext_foreach_before_tl,
5366     before .value_required:n = true,
5367     after .tl_set:N = \l__enumext_foreach_after_tl,
5368     after .value_required:n = true,
5369     start .int_set:N = \l__enumext_foreach_start_int,
5370     start .value_required:n = true,
5371     stop .int_set:N = \l__enumext_foreach_stop_int,
5372     stop .value_required:n = true,
5373     step .int_set:N = \l__enumext_foreach_step_int,
5374     step .value_required:n = true,
5375     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5376     wrapper .value_required:n = true,
5377     sep .tl_set:N = \l__enumext_foreach_sep_tl,
5378     sep .value_required:n = true,
5379     unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5380 }
5381 \keys_precompile:nnN { enumext / foreach }
5382 {
5383     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5384 }
5385 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown  $\langle keys \rangle$ .

```

5386 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5387 {
5388   \tl_if_blank:nTF {#2}
5389   {
5390     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5391   }
5392   {
5393     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5394   }
5395 }
5396 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5397 {
5398   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5399 }

```

We create the command.

```

5400 \NewDocumentCommand \foreachkeyans { +0{} m }
5401 {
5402   \__enumext_foreach_keyans:nn {#1} {#2}
5403 }

```

Finally the internal functions `\__enumext_foreach_keyans:nn` and `\__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

5404 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5405 {
5406   \tl_use:N \g__enumext_foreach_default_keys_tl
5407   \keys_set:nn { enumext / foreach } {#1}
5408   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5409   \prop_if_exist:cF { g__enumext_#2_prop }
5410   {
5411     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5412   }
5413   \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
5414   {
5415     \int_set:Nn \l__enumext_foreach_stop_int
5416     { \prop_count:c { g__enumext_#2_prop } }
5417   }
5418   \seq_clear:N \l__enumext_foreach_print_seq
5419   \int_step_function:nnnN
5420   { \l__enumext_foreach_start_int }
5421   { \l__enumext_foreach_step_int }
5422   { \l__enumext_foreach_stop_int }
5423   \__enumext_foreach_add_body:n
5424   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5425 }
5426 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5427 {
5428   \seq_put_right:Ne \l__enumext_foreach_print_seq
5429   {
5430     \exp_not:V \l__enumext_foreach_before_tl
5431     \__enumext_foreach_wrapper:n
5432     {
5433       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5434     }
5435     \exp_not:V \l__enumext_foreach_after_tl
5436   }
5437 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 17.)

### 13.51 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5438 \msg_new:nnn { enumext } { package-load }
5439 {
5440   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5441 }
5442 \msg_new:nnn { enumext } { package-not-load }
5443 {
5444   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5445 }

```

```

5446 \msg_new:nnn { enumext } { package-load-foot }
5447 {
5448   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5449 }

```

Message used in the creation of counters by **enumext** package.

```

5450 \msg_new:nnn { enumext } { counters }
5451 {
5452   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5453   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5454 }

```

Message used by **align** and **mark-pos** keys.

```

5455 \msg_new:nnn { enumext } { unknown-choice }
5456 {
5457   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5458 }

```

Message used by reserved **anskey\*** environment by **enumext** package.

```

5459 \msg_new:nnnn { enumext } { anskey-env-error }
5460 {
5461   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
5462   'enumext' ~ package, ~ It~ is~ already~ defined.
5463 }
5464 {
5465   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5466   for ~ the ~ 'save-ans' ~ key.\\
5467 }

```

Message used in the creation of *prop list* by **enumext** package.

```

5468 \msg_new:nnn { enumext } { store-prop }
5469 {
5470   * ~ Package ~ enumext: ~ Creating ~
5471   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5472 }
5473 \msg_new:nnn { enumext } { store-seq }
5474 {
5475   * ~ Package ~ enumext: ~ Creating ~
5476   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5477 }
5478 \msg_new:nnn { enumext } { store-int }
5479 {
5480   * ~ Package ~ enumext: ~ Creating ~
5481   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5482 }
5483 \msg_new:nnn { enumext } { prop-seq-int-hook }
5484 {
5485   * ~ Package ~ enumext: ~ Elements ~ in ~
5486   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5487   * ~ Package ~ enumext: ~ Elements ~ in ~
5488   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5489   * ~ Package ~ enumext: ~ Value ~ off ~
5490   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5491 }
5492 \msg_new:nnn { enumext } { item-answer-hook }
5493 {
5494   * ~ Package ~ enumext: ~ Value ~ off ~
5495   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5496   * ~ Package ~ enumext: ~ Value ~ off ~
5497   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5498   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5499 }

```

Message used by [*key* = *val*] system and **\setenumext** command.

```

5500 \msg_new:nnn { enumext } { invalid-key }
5501 {
5502   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5503 }
5504 \msg_new:nnn { enumext } { unknown-key-family }
5505 {
5506   Unknown~key~family~`\_keys_key_str'~for~enumext.
5507 }

```

Messages used in length calculation.

```

5508 \msg_new:nnn { enumext } { width-negative }
5509 {
5510   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5511   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
5512 }
5513 \msg_new:nnn { enumext } { width-zero }
5514 {
5515   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5516   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
5517 }

```

Messages used by `show-length` key in `enumext`.

```

5518 \msg_new:nnn { enumext } { list-lengths }
5519 {
5520   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
5521   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5522   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5523   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5524   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5525   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5526   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5527   \__enumext_show_length:nnn { skip } { topsep } {#1}
5528   \__enumext_show_length:nnn { skip } { parsep } {#1}
5529   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5530   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5531   ****
5532 }

```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```

5533 \msg_new:nnn { enumext } { list-lengths-not-nested }
5534 {
5535   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
5536   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5537   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5538   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5539   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5540   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5541   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5542   \__enumext_show_length:nnn { skip } { topsep } {#1}
5543   \__enumext_show_length:nnn { skip } { parsep } {#1}
5544   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5545   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5546   ****
5547 }

```

Messages used by `ref` key.

```

5548 \msg_new:nnn { enumext } { key-ref-empty }
5549 {
5550   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5551 }

```

Messages used by `save-ans` key.

```

5552 \msg_new:nnn { enumext } { save-ans-empty }
5553 {
5554   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5555 }
5556 \msg_new:nnn { enumext } { save-ans-log }
5557 {
5558   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5559 }
5560 \msg_new:nnn { enumext } { save-ans-log-hook }
5561 {
5562   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5563 }
5564 \msg_new:nnn { enumext } { save-ans-hook }
5565 {
5566   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5567 }

```

Messages used by the internal system to check answer used by `check-ans` key.

```

5568 \msg_new:nnn { enumext } { need-save-ans }
5569 {

```

```

5570     Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
5571 }
5572 \msg_new:nnn { enumext } { items-same-answer }
5573 {
5574     *****\
5575     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5576     for ~ \c_left_brace_str #2 \c_right_brace_str\
5577     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5578     'OK', ~ all ~ items ~ with ~ answer.\
5579     *****
5580 }
5581 \msg_new:nnn { enumext } { item-greater-answer }
5582 {
5583     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5584     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
5585     Items ~ > ~ Answers.
5586 }
5587 \msg_new:nnn { enumext } { item-less-answer }
5588 {
5589     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5590     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
5591     Items ~ < ~ Answers.
5592 }

```

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```

5593 \msg_new:nnn { enumext } { missing-starred }
5594 {
5595     Missing ~ '\c_backslash_str #1*' ~ #2.
5596 }
5597 \msg_new:nnn { enumext } { many-starred }
5598 {
5599     Many ~ '\c_backslash_str #1*' ~ #2.
5600 }

```

Messages used by `\printkeyans*` command.

```

5601 \msg_new:nnn { enumext } { print-starred }
5602 {
5603     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5604     #2 ~ environment ~ \msg_line_context:.
5605 }

```

Message for the nesting depth of the environment `enumext`.

```

5606 \msg_new:nnn { enumext } { list-too-deep }
5607 {
5608     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
5609     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5610 }

```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```

5611 \msg_new:nnn { enumext } { anskey-unnumber-item }
5612 {
5613     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5614 }
5615 \msg_new:nnn { enumext } { anskey-already-stored }
5616 {
5617     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5618 }
5619 \msg_new:nnn { enumext } { anskey-empty-arg }
5620 {
5621     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5622 }
5623 \msg_new:nnn { enumext } { anskey-wrong-place }
5624 {
5625     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
5626     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5627 }
5628 \msg_new:nnn { enumext } { anskey-nested }
5629 {
5630     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5631 }
5632 \msg_new:nnn { enumext } { anskey-math-mode }
5633 {
5634     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.

```

```

5635     }
5636     \msg_new:nnn { enumext } { anskey-env-wrong }
5637     {
5638         The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5639     }
5640     \msg_new:nnn { enumext } { ansPIC-wrong-place }
5641     {
5642         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5643         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5644     }
5645     \msg_new:nnn { enumext } { command-wrong-place }
5646     {
5647         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5648         '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5649     }
5650     \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5651     {
5652         The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5653         'anskey*' ~ and ~ is ~ being ~ ignored.
5654     }
5655     {
5656         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5657         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5658     }
5659     \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5660     {
5661         The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5662         'anskey*' ~ and ~ is ~ being ~ ignored.
5663     }
5664     {
5665         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5666         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5667     }
5668     \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5669     { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5670     {
5671         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5672         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5673     }
5674     \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5675     { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5676     {
5677         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5678         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5679     }

```

Messages used by [keyans](#), [keyans\\*](#) and [keyansPIC](#) environment.

```

5680     \msg_new:nnn { enumext } { keyans-nested }
5681     {
5682         The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5683     }
5684     \msg_new:nnn { enumext } { keyans-wrong-level }
5685     {
5686         Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5687         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5688     }
5689     \msg_new:nnn { enumext } { wrong-place }
5690     {
5691         Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
5692         '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5693     }
5694     \msg_new:nnn { enumext } { keyansPIC-nested }
5695     {
5696         The ~ environment ~ 'keyansPIC' ~ can't ~ be ~ nested~ \msg_line_context:~.
5697     }
5698     \msg_new:nnn { enumext } { keyansPIC-wrong-level }
5699     {
5700         Wrong ~ level ~ position ~ for ~ 'keyansPIC' ~ \msg_line_context:~ \\
5701         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5702     }
5703     \msg_new:nnn { enumext } { keyansPIC-item-cmd }
5704     {

```

```

5705     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5706 }
5707 \msg_new:nnnn { enumext } { keyans-unknown-key }
5708 {
5709     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5710     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5711 }
5712 {
5713     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5714     ~ have ~ a ~ key ~ called ~'#1'.\\
5715     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5716 }
5717 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5718 {
5719     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5720     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5721 }
5722 {
5723     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5724     ~ have ~ a ~ key ~ called ~'#1'.\\
5725     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5726 }

```

Message used by unknown *⟨keys⟩* in *enumext\**. environment.

```

5727 \msg_new:nnnn { enumext } { starred-unknown-key }
5728 {
5729     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5730     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5731 }
5732 {
5733     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5734     ~ have ~ a ~ key ~ called ~'#1'.\\
5735     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5736 }
5737 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5738 {
5739     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5740     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5741 }
5742 {
5743     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5744     ~ have ~ a ~ key ~ called ~'#1'.\\
5745     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5746 }

```

Message used by unknown *⟨keys⟩* in *enumext* environment.

```

5747 \msg_new:nnnn { enumext } { standar-unknown-key }
5748 {
5749     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5750     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5751 }
5752 {
5753     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5754     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5755     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5756 }
5757 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5758 {
5759     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5760     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5761 }
5762 {
5763     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5764     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5765     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5766 }

```

Message used by unknown *⟨keys⟩* in *\foreachkeyans*.

```

5767 \msg_new:nnnn { enumext } { for-key-unknown }
5768 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5769 {
5770     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\

```



```

5771     Check~that~you~have~spelled~the~key~name~correctly.
5772 }
5773 \msg_new:nnnn { enumext } { for-key-value-unknown }
5774 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5775 {
5776     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5777     Check~that~you~have~spelled~the~key~name~correctly.
5778 }

```

Messages used by `\getkeyans` command.

```

5779 \msg_new:nnn { enumext } { undefined-storage-anskey }
5780 {
5781     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5782 }

```

Messages used by `\miniright` command.

```

5783 \msg_new:nnn { enumext } { missing-miniright }
5784 {
5785     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5786     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5787 }
5788 \msg_new:nnn { enumext } { wrong-miniright-place }
5789 {
5790     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5791     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5792 }
5793 \msg_new:nnn { enumext } { wrong-miniright-use }
5794 {
5795     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5796     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5797 }
5798 \msg_new:nnn { enumext } { wrong-miniright-starred }
5799 {
5800     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5801 }
5802 \msg_new:nnn { enumext } { many-miniright-used }
5803 {
5804     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5805 }

```

Messages used by `\setenumextmeta` command.

```

5806 \msg_new:nnn { enumext } { unknown-set }
5807 {
5808     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5809 }
5810 \msg_new:nnn { enumext } { already-defined }
5811 {
5812     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5813 }
5814 \msg_new:nnn { enumext } { prohibited-unknown }
5815 {
5816     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5817 }

```

Messages used by `enumext*` and `keyans*` environments.

```

5818 \msg_new:nnn { enumext } { nested }
5819 {
5820     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_context:.
5821 }
5822 \msg_new:nnn { enumext } { nested-horizontal }
5823 {
5824     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~
5825 }
5826 \msg_new:nnn { enumext } { item-joined }
5827 {
5828     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5829 }
5830 \msg_new:nnn { enumext } { item-joined-columns }
5831 {
5832     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5833 }

```

13.52 Finish package

Finish package implementation.

```
5834 \file_input_stop:
5835 </package>
```

# 14 Index of Implementation

The *italic* numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>\*</code> .....	231
<code>\+</code> .....	223
<code>\-</code> .....	223
<code>\\</code> 239, 2899, 4266, 4269, 5452, 5461, 5466, 5486, 5488, 5495, 5497, 5510, 5515, 5520, 5535, 5574, 5576, 5578, 5583, 5584, 5589, 5590, 5608, 5625, 5642, 5647, 5656, 5665, 5671, 5677, 5686, 5691, 5700, 5714, 5724, 5734, 5744, 5754, 5764, 5770, 5776, 5785, 5790, 5795	
A	
above .....	<u>1713</u>
above* .....	<u>1713</u>
<code>\addvspace</code> 1283, 1311, 1354, 1357, 1525, 1528, 1625, 1631, 1666, 1672, 1693, 1699, 3724, 3885, 3903, 4157, 4161, 4514, 4529, 4575, 4589	
after .....	<u>1113</u>
align .....	<u>666</u>
<code>\Alph</code> .....	40, 45
<code>\Alph</code> .....	608, 736, 780, 846, 5160
<code>\alph</code> .....	40, 45
<code>\alph</code> .....	609, 734, 5152
<code>\anskey</code> .....	13, 79, 81, <u>2717</u>
<code>\anskey*</code> .....	14, <u>2827</u>
<code>\anspic</code> .....	16, 107, 110, <u>4171</u>
<code>\anspic*</code> .....	73
<code>\arabic</code> .....	32, 40
<code>\arabic</code> .....	607, 733, 779, 5144, 5148, 5164
B	
base-fix .....	<u>975</u>
<code>\baselineskip</code> .....	54
<code>\baselineskip</code> .....	991, 998
before .....	<u>1113</u>
before* .....	<u>1113</u>
below .....	<u>1713</u>
below* .....	<u>1713</u>
bool commands:	
<code>\bool_gset_false:N</code> 352, 353, 354, 3003, 3005, 4531, 4535, 4591	
<code>\bool_gset_true:N</code> 260, 270, 1216, 2206, 2212, 4500, 4532, 4564, 4592	
<code>\bool_if:NTF</code> . 403, 413, 430, 502, 509, 518, 525, 539, 552, 1647, 1735, 1749, 1762, 1773, 1784, 1795, 1806, 1817, 1866, 1883, 1888, 1896, 1923, 1961, 1966, 1973, 1977, 1999, 2004, 2012, 2019, 2050, 2058, 2151, 2349, 2359, 2438, 2462, 2469, 2493, 2591, 2613, 2653, 2667, 2671, 2721, 2740, 2764, 2818, 2829, 2918, 2955, 3019, 3054, 3069, 3144, 3155, 3159, 3178, 3191, 3233, 3267, 3303, 3318, 3337, 3473, 3488, 3503, 3565, 3575, 3608, 3613, 3679, 3705, 3755, 3813, 3868, 3893, 4093, 4155, 4173, 4192, 4237, 4264, 4493, 4509, 4515, 4558, 4572, 4576, 4665, 4675, 4763, 4769, 4776, 4792, 4885, 4895, 5006, 5010, 5036, 5043, 5071	
<code>\bool_if:nTF</code> 1673, 1700, 3289, 3454, 3523, 4213, 5183, 5327	
<code>\bool_if_p:N</code> 279, 294, 985, 986, 994, 995, 2030, 2031, 2039, 2040, 2164, 2190, 2203, 2204, 2209, 2210, 2526, 2536, 2548, 2563, 2564, 2598, 2639, 2640, 2941, 3131, 3132, 3169, 3170, 3652, 3654, 3665, 4220, 4221	
<code>\bool_lazy_all:nTF</code> 277, 292, 983, 2162, 2188, 2524, 2533, 2546, 2561, 3650, 3663	
<code>\bool_lazy_and:nnTF</code> 256, 266, 993, 1640, 2029, 2038, 2202, 2208, 2597, 2604, 2638, 2782, 2794, 2940, 2946, 3130	
<code>\bool_lazy_or:nnTF</code> . . 2091, 2098, 3168, 4219, 5350	
<code>\bool_new:N</code> 34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 97, 102, 103, 108, 109, 112, 131, 138, 139, 146, 153, 154, 159, 161, 162, 179, 191, 193	
<code>\bool_not_p:n</code> 257, 267, 987, 2535, 2599, 2605, 2942, 2947, 3653, 3666	
<code>\bool_set_eq:NN</code> . . . . . 3242, 3433, 4716, 4959	
<code>\bool_set_false:N</code> 410, 1005, 2136, 2137, 2169, 2174, 2178, 2182, 2195, 2882, 3627, 3772, 3821, 3908, 4046, 4090, 4634, 4660, 4713, 4901, 4956, 5196, 5197	
<code>\bool_set_true:N</code> . 284, 285, 299, 300, 395, 398, 659, 1020, 1719, 1724, 1986, 2108, 2109, 2381, 2389, 2883, 3236, 3238, 3270, 3272, 3429, 3441, 3588, 3626, 3659, 3672, 3745, 3818, 3845, 4043, 4482, 4547, 4633, 4720, 4727, 4728, 4772, 4899, 4963, 4970, 4971, 5191, 5192	
box commands:	
<code>\box_dp:N</code> . . 1571, 1572, 1575, 1582, 1595, 1603, 1609, 1617, 4104, 4109, 4157, 4248	
<code>\box_ht:N</code> . . 1354, 1357, 1368, 1369, 1380, 1382, 1397, 1400, 1408, 1409, 1420, 1422, 1437, 1440, 1447, 1448, 1459, 1461, 1476, 1479, 1525, 1528, 1536, 1537, 1545, 1546, 1558, 1560	
<code>\box_ht_plus_dp:N</code> . . . . . 4099, 4165, 4201	
<code>\box_new:N</code> . . . . . 70, 149, 150, 186, 192	
<code>\box_use_drop:N</code> . . . . . 4526, 4587, 4828, 5083	
<code>\box_wd:N</code> . . . . . 615	
C	
<code>\c</code> .....	231, 232, 882, 884, 896, 898
<code>\catcode</code> .....	2899
<code>\cB</code> .....	232
<code>\cE</code> .....	232
<code>\centering</code> .....	1675, 1702, 4292, 4519, 4580
check-ans .....	<u>2128</u>
Document class:	
<code>article</code> .....	47
clist commands:	
<code>\clist_const:Nn</code> .....	198
<code>\clist_map_function:nN</code> .....	4275
<code>\clist_map_inline:Nn</code> . 665, 930, 1112, 1127, 1208, 1729	
<code>\clist_map_inline:nn</code> . 49, 60, 78, 86, 99, 111, 141, 170, 197, 643, 696, 716, 1025, 1046, 1222, 1835, 2075, 2142, 2328, 2346, 2378, 2521, 3063, 3358, 3370, 3410, 3552, 3555, 3583, 3595, 3598, 3618, 5303	
<code>\columnbreak</code> .....	79
<code>\columnbreak</code> .....	2601
columns .....	<u>1192</u>
columns-sep .....	<u>1192</u>
<code>\columnsep</code> .....	101
<code>\columnsep</code> .....	3700, 3866
<code>\columnseprule</code> .....	101
<code>\columnseprule</code> .....	3703, 3867
Commands provide by <b>enumext</b> :	
<code>\anskey</code> 30, 69, 70, 74, 76–78, 80, 81, 88, 90, 100, 101, 119,	

128, 129, 137

\anspic\* . . . . . 30, 31, 73, 76, 88, 89, 110, 128, 129

\anspic . . . . . 30, 76, 107, 110, 137

\foreachkeyans . . . . . 133, 139

\getkeyans . . . . . 76, 128, 140

\item\* 30, 31, 73, 76, 88, 89, 91, 92, 95, 121, 126, 128, 129

\item . . . . . 91, 95, 114, 120, 122, 125, 126

\miniright . . . . . 29, 52, 59, 60, 102, 140

\printkeyans\* . . . . . 129

\printkeyans . . . . . 30, 76, 129

\setenumextmeta . . . . . 132, 140

\setenumext . . . . . 30, 129, 131, 132, 135

Counters defined by **enumext**:

enumXiii . . . . . 28, 40

enumXii . . . . . 28, 40

enumXiv . . . . . 28, 40

enumXi . . . . . 28, 40

enumXviii . . . . . 28, 40

enumXvii . . . . . 28, 40, 121

enumXvi . . . . . 28, 40

enumXv . . . . . 28, 40

cs commands:

\cs\_generate\_variant:Nn . 203, 204, 617, 633, 888, 904, 2430, 2435, 2511, 2835, 3542, 4277, 5362

\cs\_if\_exist:NTF . . . . . 587

\cs\_if\_free:NTF . . . . . 2786, 2798

\cs\_new:Nn . . . . . 217

\cs\_new:Npn . 235, 1836, 1845, 1853, 2393, 2402, 2410, 5211, 5220, 5229

\cs\_new\_eq:NN . 379, 380, 385, 386, 415, 416, 419, 420

\cs\_new\_protected:Nn . 227, 241, 249, 275, 308, 338, 344, 350, 356, 362, 370, 390, 438, 442, 459, 471, 488, 500, 516, 532, 545, 566, 756, 817, 868, 981, 1128, 1132, 1136, 1140, 1144, 1148, 1152, 1156, 1160, 1164, 1168, 1172, 1176, 1180, 1184, 1188, 1223, 1235, 1268, 1285, 1296, 1313, 1339, 1360, 1485, 1511, 1531, 1564, 1586, 1621, 1627, 1730, 1744, 1758, 1769, 1780, 1791, 1802, 1813, 1894, 1997, 2010, 2027, 2048, 2076, 2081, 2106, 2147, 2157, 2200, 2215, 2222, 2231, 2236, 2241, 2246, 2255, 2260, 2265, 2436, 2460, 2467, 2491, 2498, 2512, 2738, 2757, 2773, 2836, 2872, 2903, 2938, 2980, 3001, 3009, 3052, 3067, 3095, 3128, 3164, 3176, 3189, 3275, 3285, 3296, 3312, 3328, 3450, 3466, 3482, 3496, 3619, 3648, 3677, 3684, 3714, 3731, 3753, 3775, 3811, 3835, 3852, 3877, 3891, 3912, 4065, 4259, 4273, 4278, 4302, 4312, 4343, 4472, 4491, 4537, 4556, 4620, 4647, 4654, 4663, 4673, 4698, 4839, 4883, 4914, 4920, 4941, 4998, 5094

\cs\_new\_protected:Npn 205, 209, 213, 423, 585, 602, 612, 618, 737, 781, 851, 875, 889, 1657, 1686, 1862, 1881, 1951, 1984, 2086, 2270, 2347, 2357, 2379, 2387, 2422, 2431, 2587, 2650, 2665, 2703, 2707, 2827, 2858, 2862, 2893, 3029, 3105, 3149, 3229, 3248, 3371, 3375, 3389, 3393, 3411, 3415, 3425, 3437, 3511, 3545, 3586, 3630, 3831, 4074, 4081, 4088, 4190, 4209, 4233, 4374, 4423, 4637, 4704, 4711, 4725, 4733, 4738, 4748, 4907, 4947, 4954, 4968, 4976, 4993, 5116, 5129, 5177, 5300, 5312, 5336, 5348, 5386, 5396, 5404, 5426

\cs\_new\_protected\_nopar:Nn . . 3973, 4015, 4023, 4031, 4683, 4691, 4822, 4926, 4934, 5077

\cs\_new\_protected\_nopar:Npn . . 3965, 3981, 4754, 4800, 5026, 5051

\cs\_set:Npn . . . . . 2522, 2559, 5122

\cs\_set\_eq:NN . . 4610, 4611, 4802, 4872, 4873, 5053

\cs\_set\_protected:Nn . . . 1051, 1067, 1080, 1092

\cs\_set\_protected:Npn 45, 54, 71, 79, 94, 100, 134, 166, 177, 634, 644, 666, 701, 717, 763, 905, 931, 1007, 1030, 1104, 1113, 1192, 1209, 1713, 1824, 2067, 2128, 2287, 2329, 2365, 2514, 3056, 3347, 3363, 3403, 3543, 3584

\cs\_to\_str:N . . . . . 604, 627

\cs\_undefine:N . . . . . 2775, 2776, 2777, 2778

## D

\d . . . . . 223

\DeclareDocumentEnvironment . . . . . 570

dim commands:

\dim\_abs:n . . . . . 3516, 3521

\dim\_add:Nn . . . . . 4108, 4337, 4368

\dim\_compare:nNnTF . . 1053, 1069, 1082, 1094, 1372, 1384, 1412, 1424, 1451, 1463, 1540, 1548, 1659, 1688, 3513, 3518, 3524, 3530, 3532, 3534, 3689, 3736, 3839, 3856, 4083, 4314, 4330, 4345, 4361, 4474, 4539

\dim\_compare:nTF . . . . . 2623, 2968, 3778, 3915

\dim\_eval:n . . . . . 991, 4163, 4244

\dim\_gset\_eq:NN . . . . . 4483, 4548

\dim\_gzero:N . . . . . 3007, 4534, 4594

\dim\_new:N . 67, 74, 75, 76, 96, 143, 151, 152, 185, 187, 188, 194

\dim\_set:Nn . 615, 1021, 3265, 3516, 3521, 3523, 3526, 3527, 3531, 3533, 3536, 3537, 3539, 3692, 3739, 3777, 3841, 3858, 3914, 4097, 4199, 4280, 4316, 4323, 4347, 4354, 4409, 4458, 4476, 4541, 4750

\dim\_set\_eq:NN 724, 770, 839, 843, 3180, 3181, 3193, 3194, 3260, 3554, 3597, 3700, 3866, 4416, 4419, 4420, 4465, 4468, 4469, 4743, 4814, 5065

\dim\_sub:Nn . . . . . 3783, 3920, 4332, 4363

\dim\_use:N . 1054, 1062, 1660, 1670, 2501, 2504, 2509, 3280, 3282, 3333, 3690, 3694, 3695, 3697, 3737, 3742, 3743, 3749, 3780, 3785

\dim\_zero:N . . . . . 3589, 3703, 3867, 4110

\dim\_zero\_new:N . . . . . 584

\c\_zero\_dim 1056, 1070, 1083, 1095, 1660, 1688, 2625, 2970, 3513, 3518, 3524, 3531, 3690, 3737, 3780, 3839, 3856, 3917, 4083, 4314, 4330, 4345, 4361, 4474, 4539

\dimeval . . . . . 2294

## E

\end . . . 2464, 2495, 3721, 3882, 4147, 4294, 5185, 5195, 5203

end internal commands:

\end\_\_enumext\_mini\_page . 1668, 1695, 3764, 3902, 4498, 4562, 4588

\endgroup . . . . . 2899

\endlist . . . . . 380

\endminipage . . . . . 386

enumext . . . . . 5, 3789

enumext internal commands:

\l\_\_enumext\_\_ref\_the\_count\_tl . . . . . 43

\l\_\_enumext\_\_resume\_name\_tl . . . . . 65

\\_\_enumext\_add\_meta\_key:nnn . . 133, 5314, 5330, 5331, 5333, 5336

\\_\_enumext\_add\_pre\_parsep: . 52, 1233, 1235, 1235

\\_\_enumext\_after\_args\_exec: 50, 1128, 1140, 3802

\\_\_enumext\_after\_args\_exec\_v: 1144, 1156, 3935

\\_\_enumext\_after\_args\_exec\_viii: . . 1160, 1184

\\_\_enumext\_after\_args\_exec\_viii: . . . . . 1188

\\_\_enumext\_after\_env:nn 84, 86, 87, 103, 116, 123, 209, 209, 558, 562, 2913, 3807, 4507, 4570, 4855

\\_\_enumext\_after\_hyperref: . . 36, 388, 388, 390

\l\_\_enumext\_after\_list\_args\_v\_tl . . . . . 1158

\l\_\_enumext\_after\_list\_args\_viii\_tl 1186, 4820

`\l__enumext_after_list_args_viii_tl` .. 1190, 5075  
`\__enumext_after_list_vii:` 116, 119, 4618, 4654, 4654  
`\__enumext_after_list_viii:` ... 125, 4881, 4920, 4920  
`\__enumext_after_stop_list:` 50, 102, 1128, 1136, 3769  
`\__enumext_after_stop_list_v:` 1144, 1152, 3909  
`\l__enumext_after_stop_list_v_tl` ..... 1154  
`\__enumext_after_stop_list_vii:` .. 119, 1160, 1176, 4657  
`\l__enumext_after_stop_list_vii_tl` ... 1178  
`\__enumext_after_stop_list_viii:` . 1180, 4923  
`\l__enumext_after_stop_list_viii_tl` ... 1182  
`\l__enumext_align_label_pos_v_str` .... 3500  
`\l__enumext_align_label_pos_X_str` ..... 79  
`\l__enumext_align_label_vii_str` ..... 4789  
`\l__enumext_align_label_viii_str` ..... 5040  
`\l__enumext_align_label_X_str` ..... 177  
`\c__enumext_all_envs_clist` . 198, 665, 930, 1112, 1127, 1208, 1729  
`\c__enumext_all_families_seq` .. 131, 5268, 5294  
`\l__enumext_anskey_env_bool` 33, 84, 34, 285, 300, 2829  
`\__enumext_anskey_env_clean_vars:` . 87, 2934, 2938, 3001  
`\__enumext_anskey_env_define_keys:` 84, 2827, 2836, 2907  
`\__enumext_anskey_env_exec:` 85, 2832, 2903, 2903  
`\__enumext_anskey_env_make:n` 69, 84, 2111, 2827, 2827, 2835  
`\__enumext_anskey_env_reset_keys:` 84, 86, 2827, 2872, 2935  
`\__enumext_anskey_env_save_keys:` .. 86, 2915, 2938, 2938  
`\__enumext_anskey_env_store:` .. 86, 2931, 2938, 2980  
`\__enumext_anskey_env_unknown:n` 84, 2827, 2855, 2858  
`\__enumext_anskey_env_unknown:nn` . 2827, 2860, 2862  
`\l__enumext_anskey_level_int` .. 28, 2759, 2760  
`\__enumext_anskey_safe_inner:` . 82, 2732, 2738, 2757  
`\__enumext_anskey_safe_inner:n` ..... 81  
`\__enumext_anskey_safe_outer:` . 81, 2719, 2738, 2738  
`\__enumext_anskey_show_wrap_arg:n` . 80, 2650, 2650, 2669, 2684  
`\__enumext_anskey_show_wrap_left:n` 80, 2595, 2665, 2665  
`\__enumext_anskey_unknown:n` 81, 2687, 2701, 2703  
`\__enumext_anskey_unknown:nn` . 2687, 2705, 2707  
`\__enumext_anskey_wrapper:n` ..... 2291, 2663  
`\l__enumext_anspic_above_int` . 142, 4281, 4282, 4284  
`\__enumext_anspic_args:nnn` 110, 112, 4171, 4187, 4259  
`\l__enumext_anspic_args_seq` 110, 112, 142, 4185, 4293, 4306  
`\l__enumext_anspic_below_int` . 142, 4281, 4282, 4285  
`\l__enumext_anspic_body_box` ... 142, 4198, 4201  
`\__enumext_anspic_body_dim:n` .. 111, 4171, 4190, 4236  
`\l__enumext_anspic_body_htdp_dim` .. 111, 142, 4199, 4247  
`\__enumext_anspic_exec:` ..... 4171  
`\__enumext_anspic_exec:` ..... 112, 4142, 4302  
`\__enumext_anspic_label:nn` 111, 4171, 4209, 4239, 4254  
`\l__enumext_anspic_label_above_bool` ... 142, 4043, 4046, 4093, 4155, 4192, 4237, 4264  
`\l__enumext_anspic_label_box` .. 142, 4096, 4099  
`\l__enumext_anspic_label_htdp_dim` . 109, 142, 4097, 4103, 4165, 4246  
`\__enumext_anspic_label_pos:nnn` .. 111, 4171, 4233, 4262  
`\l__enumext_anspic_label_sep_skip` 4053, 4105, 4166, 4249, 4266  
`\l__enumext_anspic_layout_style_tl` 4055, 4304, 4309  
`\l__enumext_anspic_mini_pos_str` .. 142, 4044, 4047, 4291  
`\l__enumext_anspic_mini_width_dim` 142, 4211, 4280, 4291  
`\__enumext_anspic_print:n` 112, 4171, 4273, 4277, 4306, 4309  
`\__enumext_anspic_row:n` .. 112, 4171, 4275, 4278  
`\__enumext_anspic_start_list_tag:` 3989, 4015, 4261  
`\__enumext_anspic_stop_list_tag:` . 3989, 4031, 4271  
`\__enumext_anspic_stop_start_list_tag:` 3989, 4023, 4263  
`\__enumext_at_begin_document:n` .. 35, 205, 205, 377, 383  
`\l__enumext_base_line_fix_bool` 47, 48, 130, 977, 986, 1005, 5191, 5196  
`\__enumext_before_args_exec:` 50, 102, 119, 1128, 1128, 3734  
`\__enumext_before_args_exec_v:` 1144, 1144, 3838  
`\__enumext_before_args_exec_vii:` . 1160, 1160, 4651  
`\__enumext_before_args_exec_viii:` 1164, 4917  
`\__enumext_before_env:nn` 84, 209, 213, 2780, 2792, 2804, 2905  
`\__enumext_before_keys_exec:` .. 50, 1128, 1132, 3799  
`\__enumext_before_keys_exec_v:` 1144, 1148, 3932  
`\__enumext_before_keys_exec_vii` ..... 1160  
`\__enumext_before_keys_exec_vii:` . 1168, 4605  
`\__enumext_before_keys_exec_viii:` 1172, 4867  
`\__enumext_before_list:` .. 102, 3731, 3731, 3793  
`\__enumext_before_list_v:` ... 3835, 3835, 3927  
`\__enumext_before_list_vii:` ... 119, 4600, 4647, 4647  
`\__enumext_before_list_viii:` . 124, 4863, 4914, 4914  
`\l__enumext_before_no_starred_key_v_tl` 1150  
`\l__enumext_before_no_starred_key_vii_-tl` ..... 1170  
`\l__enumext_before_no_starred_key_viii_-tl` ..... 1174  
`\l__enumext_before_starred_key_v_tl` ... 1146  
`\l__enumext_before_starred_key_vii_tl` . 1162  
`\l__enumext_before_starred_key_viii_tl` 1166  
`\__enumext_calc_hspace:NNNNNNN` 97, 3511, 3511,

3542, 3547, 3590  
 \\_\_enumext\_check\_ans\_active: 70, 102, 119, 2147, 2147, 3735, 4650  
 \g\_\_enumext\_check\_ans\_item\_tl . . . . . 89  
 \g\_\_enumext\_check\_ans\_key\_bool 71, 72, 153, 352, 2206, 2212, 3019  
 \l\_\_enumext\_check\_ans\_key\_bool 71, 2132, 2137, 2203, 2209  
 \\_\_enumext\_check\_ans\_key\_hook: . . 71, 102, 119, 2200, 2200, 3770, 4658  
 \\_\_enumext\_check\_ans\_level: 70, 2147, 2153, 2157  
 \\_\_enumext\_check\_ans\_log: 72, 87, 2246, 2246, 3023  
 \\_\_enumext\_check\_ans\_log\_msg\_greater: 2246, 2252, 2265  
 \\_\_enumext\_check\_ans\_log\_msg\_less: 2246, 2250, 2255  
 \\_\_enumext\_check\_ans\_log\_msg\_same\_ok: 2246, 2251, 2260  
 \\_\_enumext\_check\_ans\_msg\_greater: 2222, 2228, 2241  
 \\_\_enumext\_check\_ans\_msg\_less: 2222, 2226, 2231  
 \\_\_enumext\_check\_ans\_msg\_same\_ok: 2222, 2227, 2236  
 \\_\_enumext\_check\_ans\_show: 71, 72, 87, 2222, 2222, 3021  
 \l\_\_enumext\_check\_answers\_bool 69, 70, 81, 91, 92, 153, 2109, 2136, 2151, 2438, 2462, 2469, 2493, 2721, 2918, 3144, 3233, 3267, 4769  
 \\_\_enumext\_check\_starred\_cmd:n 34, 73, 89, 123, 2270, 2270, 3938, 4153, 4880  
 \g\_\_enumext\_check\_starred\_cmd\_int . . 95, 153, 2273, 2279, 2284, 3448, 4218, 5005  
 \l\_\_enumext\_check\_start\_line\_env\_tl . 34, 153, 315, 323, 331, 2276, 2282, 2285  
 \l\_\_enumext\_columns\_sep\_v\_dim 3856, 3858, 3866  
 \l\_\_enumext\_columns\_sep\_vii\_dim . . 4314, 4316, 4325, 4337, 4413, 4836  
 \l\_\_enumext\_columns\_sep\_viii\_dim . 4345, 4347, 4356, 4368, 4462, 5091  
 \l\_\_enumext\_columns\_v\_int 1505, 1523, 1691, 3854, 3862, 3874, 3879  
 \l\_\_enumext\_columns\_vii\_int . . 4319, 4322, 4326, 4335, 4377, 4381, 4384, 4390, 4396, 4400, 4830, 4844  
 \l\_\_enumext\_columns\_viii\_int . 4350, 4353, 4357, 4366, 4426, 4430, 4433, 4439, 4445, 4449, 5085, 5100  
 \l\_\_enumext\_counter\_i\_tl . . . . . 45, 594  
 \l\_\_enumext\_counter\_ii\_tl . . . . . 45, 595  
 \l\_\_enumext\_counter\_iii\_tl . . . . . 45, 596  
 \l\_\_enumext\_counter\_iv\_tl . . . . . 45, 597  
 \c\_\_enumext\_counter\_style\_tl . . . . . 32, 50, 229  
 \g\_\_enumext\_counter\_styles\_tl . 28, 40, 67, 605, 623  
 \l\_\_enumext\_counter\_v\_tl . . . . . 45, 598, 859  
 \l\_\_enumext\_counter\_vi\_tl . . . . . 45, 599  
 \l\_\_enumext\_counter\_vii\_tl . . . . . 45, 600, 791  
 \l\_\_enumext\_counter\_viii\_tl . . . . . 45, 601, 807  
 \l\_\_enumext\_current\_widest\_dim 28, 67, 629, 725, 771, 840, 844  
 \\_\_enumext\_def\_meta\_key:nnn . . 133, 5314, 5342, 5348, 5362  
 \\_\_enumext\_default\_item:n . . . 3229, 3229, 3293  
 \\_\_enumext\_define\_counters:Nn 28, 585, 585, 594, 595, 596, 597, 598, 599, 600, 601  
 \\_\_enumext\_endminipage: . 36, 377, 386, 579, 4528, 4824, 5079  
 \g\_\_enumext\_envir\_name\_tl 33, 34, 286, 301, 360, 2079, 2084, 2094, 2234, 2239, 2244, 2258, 2263, 2268  
 \l\_\_enumext\_envir\_name\_tl . 33, 34, 34, 255, 265, 314, 322, 330, 5710, 5713, 5720, 5723, 5730, 5733, 5740, 5743, 5749, 5753, 5759, 5763, 5820, 5824  
 \\_\_enumext\_execute\_after\_env: 34, 35, 68, 72, 83, 87, 3009, 3009, 3809, 4857  
 \\_\_enumext\_fake\_item\_indent: . 1051, 1051, 3574  
 \l\_\_enumext\_fake\_item\_indent\_v\_dim 1070, 1075  
 \l\_\_enumext\_fake\_item\_indent\_v\_tl 1072, 3430, 3434, 3442  
 \\_\_enumext\_fake\_item\_indent\_vii: . 1051, 1080, 3607  
 \l\_\_enumext\_fake\_item\_indent\_vii\_dim . 1083, 1087  
 \l\_\_enumext\_fake\_item\_indent\_vii\_tl . . 1085, 4819  
 \\_\_enumext\_fake\_item\_indent\_viii: 1051, 1092, 3612  
 \l\_\_enumext\_fake\_item\_indent\_viii\_dim 1095, 1099  
 \l\_\_enumext\_fake\_item\_indent\_viii\_tl . 1097, 5070  
 \l\_\_enumext\_fake\_item\_indent\_X\_tl . . . . 100  
 \\_\_enumext\_fake\_make\_label\_vii:n . 121, 4754, 4754, 4811  
 \\_\_enumext\_fake\_make\_label\_viii:n 5026, 5026, 5062  
 \\_\_enumext\_filter\_first\_level:n . . 130, 5211, 5211, 5245, 5256  
 \\_\_enumext\_filter\_first\_level\_key:n 131, 5211, 5216, 5220  
 \\_\_enumext\_filter\_first\_level\_pair:nn . 131, 5211, 5217, 5229  
 \\_\_enumext\_filter\_save\_key:n . . 75, 2354, 2362, 2385, 2391, 2393, 2393, 5142, 5146, 5150, 5154, 5158, 5162  
 \\_\_enumext\_filter\_save\_key\_key:n . . 75, 2393, 2398, 2402  
 \\_\_enumext\_filter\_save\_key\_pair:nn 75, 2393, 2399, 2410  
 \\_\_enumext\_filter\_series:n 64, 1836, 1836, 1874, 1886, 1891  
 \\_\_enumext\_filter\_series\_key:n 64, 1836, 1841, 1845  
 \\_\_enumext\_filter\_series\_pair:nn . . 64, 1836, 1842, 1853  
 \\_\_enumext\_first\_item\_tmp\_vii: 118, 120, 4610, 4683, 4683  
 \\_\_enumext\_first\_item\_tmp\_viii: . . 125, 4872, 4926, 4926  
 \g\_\_enumext\_footnote\_standar\_arg\_seq . . 171, 455, 465, 468  
 \g\_\_enumext\_footnote\_standar\_int 171, 449, 452, 454, 456  
 \g\_\_enumext\_footnote\_standar\_int\_seq . . 171, 456, 461, 464, 469  
 \g\_\_enumext\_footnote\_starred\_arg\_seq . . 171, 484, 494, 497  
 \g\_\_enumext\_footnote\_starred\_int 171, 478, 481, 483, 485  
 \g\_\_enumext\_footnote\_starred\_int\_seq . . 171, 485, 490, 493, 498  
 \\_\_enumext\_footnotes\_key\_bool . . . . . 36



`\l__enumext_footnotes_key_bool` 31, 36, 161, 398, 403, 410, 509, 525, 539, 552  
`\__enumext_footnotetext:nn` .. 438, 438, 466, 495  
`\__enumext_foreach_add_body:n` 134, 5363, 5423, 5426  
`\l__enumext_foreach_after_tl` ..... 5367, 5435  
`\l__enumext_foreach_before_tl` .... 5365, 5430  
`\g__enumext_foreach_default_keys_tl` 133, 126, 5385, 5406  
`\__enumext_foreach_keyans:nn` . 134, 5363, 5402, 5404  
`\l__enumext_foreach_name_prop_tl` . 126, 5408, 5433  
`\l__enumext_foreach_print_seq` 126, 5418, 5424, 5428  
`\l__enumext_foreach_sep_tl` ..... 5377, 5424  
`\l__enumext_foreach_start_int` .... 5369, 5420  
`\l__enumext_foreach_step_int` ..... 5373, 5421  
`\l__enumext_foreach_stop_int` . 5371, 5413, 5415, 5422  
`\__enumext_foreach_wrapper:n` ..... 5375, 5431  
`\__enumext_getkeyans:nn` .. 129, 5111, 5125, 5129  
`\__enumext_getkeyans_aux:n` 128, 5111, 5113, 5116  
`\l__enumext_hyperref_bool` 31, 36, 161, 395, 413, 430, 2640, 3132, 4763  
`\__enumext_hypertarget:nn` 36, 388, 415, 419, 435  
`\__enumext_if_is_int:n` ..... 221  
`\__enumext_if_is_int:nTF` ..... 221, 877, 891  
`\__enumext_internal_mini_page:` 39, 99, 118, 566, 566, 3622, 4623  
`\__enumext_is_not_nested:` 28, 33, 99, 118, 249, 249, 3621, 4622  
`\__enumext_is_on_first_level:` . 28, 33, 99, 118, 249, 275, 3628, 4635  
`\g__enumext_item_anskey_int` 81, 89, 153, 347, 374, 375, 2219, 2589, 3146  
`\__enumext_item_answer_diff:` .. 71, 72, 87, 2215, 2215, 3016  
`\g__enumext_item_answer_diff_int` . 71, 72, 153, 348, 2217, 2224, 2248  
`\l__enumext_item_column_pos_vii_int` 120, 4384, 4390, 4396, 4400, 4407, 4694, 4830, 4833  
`\l__enumext_item_column_pos_viii_int` .. 125, 4433, 4439, 4445, 4449, 4456, 4937, 5085, 5088  
`\l__enumext_item_column_pos_X_int` ..... 177  
`\g__enumext_item_count_all_vii_int` 120, 4408, 4695, 4844, 4852  
`\g__enumext_item_count_all_viii_int` 125, 4457, 4938, 5099, 5108  
`\g__enumext_item_count_all_X_int` ..... 177  
`\g__enumext_item_number_bool` ..... 153  
`\l__enumext_item_number_bool` 70, 159, 2169, 2174, 2178, 2182, 2195, 2764, 2818, 3236, 3270, 4772  
`\g__enumext_item_number_int` 70, 71, 153, 346, 373, 375, 2168, 2173, 2177, 2181, 2194, 2219, 3235, 3269, 4771  
`\__enumext_item_peek_args_vii:` 120, 4691, 4696, 4698  
`\__enumext_item_peek_args_viii:` .. 125, 4934, 4939, 4941  
`\__enumext_item_star_exec:` 92, 3248, 3275, 3320, 3339  
`\l__enumext_item_starred_vii_bool` 4713, 4727, 4776  
`\l__enumext_item_starred_viii_bool` 4956, 4970, 5036, 5071  
`\l__enumext_item_starred_X_bool` ..... 177  
`\__enumext_item_std:w` 35, 91, 92, 95, 377, 381, 3239, 3245, 3273, 3430, 3434, 3442  
`\g__enumext_item_symbol_aux_tl` . 92, 130, 3253, 3256, 3281, 3325, 3343  
`\g__enumext_item_symbol_aux_vii_tl` 4735, 4778, 4781, 4785, 4787  
`\g__enumext_item_symbol_aux_X_tl` ..... 177  
`\l__enumext_item_symbol_sep_vii_dim` .. 4743, 4750, 4784, 4786  
`\l__enumext_item_symbol_vii_tl` ..... 4781  
`\l__enumext_item_text_vii_box` .... 4803, 4828  
`\l__enumext_item_text_viii_box` ... 5054, 5083  
`\l__enumext_item_text_X_box` ..... 177  
`\l__enumext_item_width_vii_dim` ... 4323, 4332, 4411, 4419, 4420  
`\l__enumext_item_width_viii_dim` .. 4354, 4363, 4460, 4468, 4469  
`\l__enumext_item_width_X_dim` ..... 177  
`\l__enumext_itemindent_X_dim` ..... 71  
`\l__enumext_itemsep_i_skip` ... 1366, 1373, 1376, 1378, 1385, 1389, 1392, 1394, 1534, 1541, 1543, 1544, 1549, 1553, 1555, 1556  
`\l__enumext_itemsep_ii_skip` .. 1406, 1413, 1416, 1418, 1425, 1429, 1432, 1434  
`\l__enumext_itemsep_iii_skip` . 1445, 1452, 1455, 1457, 1464, 1468, 1471, 1473  
`\l__enumext_itemsep_vii_skip` ..... 4850  
`\l__enumext_itemsep_viii_skip` ..... 5106  
`\l__enumext_joined_item_aux_vii_int` .. 4405, 4406, 4407, 4408, 4414  
`\l__enumext_joined_item_aux_viii_int` . 4454, 4455, 4456, 4457, 4463  
`\l__enumext_joined_item_aux_X_int` .... 177  
`\__enumext_joined_item_vii:w` .. 120, 4691, 4701, 4702, 4704  
`\l__enumext_joined_item_vii_int` .. 4376, 4377, 4380, 4382, 4388, 4393, 4398, 4403, 4405, 4411  
`\__enumext_joined_item_viii:w` . 125, 4934, 4944, 4945, 4947  
`\l__enumext_joined_item_viii_int` . 4425, 4426, 4429, 4431, 4437, 4442, 4447, 4452, 4454, 4460  
`\l__enumext_joined_item_X_int` ..... 177  
`\l__enumext_joined_width_vii_dim` . 4409, 4416, 4419, 4805, 4813  
`\l__enumext_joined_width_viii_dim` 4458, 4465, 4468, 5056, 5064  
`\l__enumext_joined_width_X_dim` ..... 177  
`\__enumext_keyans_addto_prop:n` 87, 3029, 3029, 3445, 4215  
`\__enumext_keyans_addto_seq:n` . 89, 3105, 3105, 3447, 4217  
`\__enumext_keyans_addto_seq_link:` 3105, 3126, 3128, 5004  
`\__enumext_keyans_default_item:n` .. 95, 3425, 3425, 3462  
`\l__enumext_keyans_env_bool` 34, 3653, 3666, 3818, 3908  
`\__enumext_keyans_fake_item_indent:` .. 1051, 1067, 3564  
`\l__enumext_keyans_level_h_int` .. 124, 28, 800, 826, 2748, 2810, 3083, 4629, 4889, 4890



```

\l__enumext_keyans_level_int .. 28, 1651, 2744,
    2806, 3078, 3817, 3822, 4181
\__enumext_keyans_make_label: . 96, 3466, 3466,
    3562
\__enumext_keyans_make_label_box: 3466, 3470,
    3475, 3496
\__enumext_keyans_make_label_std: 3466, 3478,
    3482
\__enumext_keyans_mini_right_cmd:n 60, 1653,
    1686, 1686
\__enumext_keyans_mini_set_vskip: ..... 57
\__enumext_keyans_minipage_add_space: 1485,
    1511, 3847
\__enumext_keyans_minipage_set_skip: . 1485,
    1485, 1513
\__enumext_keyans_multi_addvspace: 1285, 1296,
    3871
\__enumext_keyans_multi_set_vskip: 53, 1285,
    1285, 1298
\__enumext_keyans_multicols_start: 3835, 3850,
    3852
\__enumext_keyans_multicols_stop: 1690, 3835,
    3877, 3906
\__enumext_keyans_name_and_start: 28, 34, 124,
    308, 308, 3819, 4072, 4894
\__enumext_keyans_parse_keys:n 3831, 3831, 3926
\__enumext_keyans_pic_arg_two: 109, 4065, 4088,
    4118
\l__enumext_keyans_pic_level_int .. 28, 1635,
    2752, 2814, 3032, 3073, 3108, 3196, 4067, 4068
\__enumext_keyans_pic_parse_keys:n 4065, 4074,
    4117
\g__enumext_keyans_pic_parsep_skip ... 142
\__enumext_keyans_pic_safe_exec: . 108, 4065,
    4065, 4116
\__enumext_keyans_pic_skip_abs:N . 108, 4065,
    4081, 4092
\__enumext_keyans_pre_itemsep_skip: .. 1485,
    1504, 1531
\__enumext_keyans_redefine_item: .. 96, 3450,
    3450, 3561
\__enumext_keyans_ref: ..... 45, 851, 868, 3563
\__enumext_keyans_ref:n ..... 45, 848, 851, 851
\__enumext_keyans_safe_exec: . 3811, 3811, 3925
\__enumext_keyans_set_item_width: 105, 3912,
    3912, 3934
\__enumext_keyans_show_ans: .. 3149, 3157, 3176
\__enumext_keyans_show_item_opt: .. 95, 3149,
    3164, 3443, 4230, 5073
\__enumext_keyans_show_left:n . 95, 3149, 3149,
    3440, 4224
\__enumext_keyans_show_pos: .. 3149, 3161, 3189
\__enumext_keyans_starred_item:n .. 95, 3437,
    3437, 3458
\__enumext_keyans_store_ref: .. 88, 3052, 3052,
    3446, 4216, 5002
\__enumext_keyans_store_ref_aux_i: 88, 3052,
    3064, 3067
\__enumext_keyans_store_ref_aux_ii: 89, 3052,
    3093, 3095
\__enumext_keyans_unknown_keys:n . 3363, 3367,
    3371, 4063
\__enumext_keyans_unknown_keys:nn 3363, 3373,
    3375
\__enumext_keyans_wrapper_opt:n .. 2297, 3172

\l__enumext_label_copy_i_tl .. 2555, 3071, 3076,
    3081, 3086
\l__enumext_label_copy_v_tl ..... 3081
\l__enumext_label_copy_vi_tl ..... 3076
\l__enumext_label_copy_vii_tl 2531, 2542, 2571,
    3071
\l__enumext_label_copy_viii_tl ..... 3086
\l__enumext_label_copy_X_tl ..... 163
\l__enumext_label_fill_left_v_tl ..... 3486
\l__enumext_label_fill_left_X_tl ..... 100
\l__enumext_label_fill_right_v_tl .... 3493
\l__enumext_label_fill_right_X_tl .... 100
\l__enumext_label_font_style_v_tl 3487, 3502,
    4228
\l__enumext_label_font_style_vii_tl ... 4791
\l__enumext_label_font_style_viii_tl .. 5042
\l__enumext_label_i_tl ..... 717
\l__enumext_label_ii_tl ..... 717
\l__enumext_label_iii_tl ..... 717
\l__enumext_label_iv_tl ..... 717
\__enumext_label_style:Nnn 28, 40, 618, 618, 633,
    722, 768, 837, 841
\l__enumext_label_v_tl 89, 834, 3037, 3113, 3183,
    3223, 3439, 3444, 3929, 4096, 4223, 4225
\l__enumext_label_vi_tl 89, 834, 3034, 3110, 4223,
    4225, 4229
\l__enumext_label_vii_tl . 763, 4722, 4745, 4752
\l__enumext_label_viii_tl 763, 4965, 4996, 5000
\l__enumext_label_width_by_box .. 67, 614, 615
\__enumext_label_width_by_box:Nn 40, 612, 612,
    617, 629, 901
\l__enumext_labelsep_i_dim ... 3181, 3186, 3194,
    3226, 5008, 5023
\l__enumext_labelsep_v_dim ..... 3861
\l__enumext_labelsep_vii_dim . 2656, 3181, 3194,
    4318, 4328, 4412, 4687, 4743, 4798, 4807
\l__enumext_labelsep_viii_dim 4349, 4359, 4461,
    4930, 5049, 5058
\l__enumext_labelwidth_i_dim . 3180, 3186, 3193,
    3226, 5008, 5023
\l__enumext_labelwidth_v_dim ..... 3500, 3861
\l__enumext_labelwidth_vii_dim ... 2656, 3180,
    3193, 4318, 4327, 4412, 4687, 4789, 4806
\l__enumext_labelwidth_viii_dim .. 4349, 4358,
    4461, 4930, 5040, 5057
\l__enumext_leftmargin_tmp_v_bool . 109, 4090
\l__enumext_leftmargin_tmp_X_bool ..... 71
\l__enumext_leftmargin_tmp_X_dim ..... 71
\l__enumext_leftmargin_X_dim ..... 71
\__enumext_level: 217, 217, 746, 749, 750, 758, 760,
    1054, 1058, 1062, 1130, 1134, 1138, 1142, 1225, 1227,
    1229, 1231, 1273, 1275, 1277, 1279, 1283, 1317, 1323,
    1328, 1330, 1333, 1336, 1349, 1352, 1660, 1664, 1670,
    1733, 1735, 1737, 1740, 1747, 1749, 1751, 1754, 2349,
    2351, 2353, 2381, 2382, 2384, 2440, 2448, 2452, 2456,
    2660, 2661, 3238, 3239, 3243, 3244, 3245, 3253, 3261,
    3262, 3265, 3272, 3273, 3277, 3280, 3282, 3316, 3317,
    3318, 3321, 3324, 3333, 3334, 3336, 3337, 3340, 3659,
    3672, 3679, 3687, 3690, 3692, 3694, 3695, 3696, 3697,
    3700, 3705, 3711, 3717, 3724, 3737, 3739, 3742, 3743,
    3745, 3749, 3755, 3780, 3785, 3796, 3798
\l__enumext_level_h_int 118, 28, 258, 281, 295, 784,
    819, 1642, 2165, 2185, 2550, 2784, 2796, 3667, 4624,
    4625

```

```

\l__enumext_level_int . 99, 28, 219, 268, 280, 296,
    568, 1237, 1362, 1641, 2159, 2191, 2527, 2537, 2543,
    2549, 2556, 2565, 2570, 2783, 2795, 3011, 3578, 3623,
    3624, 3635, 3643, 3657, 3670, 3701, 3826, 4177, 4667,
    4677, 4902, 5750, 5754, 5760, 5764
\__enumext_list_arg_two_i: . . . . . 3543
\__enumext_list_arg_two_ii: . . . . . 3543
\__enumext_list_arg_two_iii: . . . . . 3543
\__enumext_list_arg_two_iv: . . . . . 3543
\__enumext_list_arg_two_v: . 96, 3543, 3931, 4091
\__enumext_list_arg_two_vii: . . . . 3584, 4604
\__enumext_list_arg_two_viii: . . . . 3584, 4866
\l__enumext_listoffset_v_dim . 3863, 3917, 3920
\l__enumext_listparindent_vii_dim 4814, 4818
\l__enumext_listparindent_viii_dim 5065, 5069
\__enumext_log_answer_vars: . 35, 362, 370, 3018
\__enumext_log_global_vars: . 35, 362, 362, 3017
\__enumext_make_label: . . . . 93, 3296, 3296, 3572
\__enumext_make_label_box: . . . 3296, 3300, 3305,
    3328
\__enumext_make_label_std: . . . 3296, 3308, 3312
\l__enumext_mark_answer_sym_tl 77, 2303, 2506,
    2673, 3198, 3211, 5012
\l__enumext_mark_position_str 130, 2307, 2308,
    2334, 2335, 2504
\l__enumext_mark_ref_sym_tl . 2320, 2645, 3140
\l__enumext_meta_path_tl . 126, 5338, 5339, 5341,
    5342
\c__enumext_meta_paths_prop . . . . . 132, 5314
\__enumext_mini_addvspace_vii: 59, 1621, 1621,
    4486
\__enumext_mini_addvspace_viii: 59, 1621, 1627,
    4551
__enumext_mini_env* . . . . . 566
\__enumext_mini_page 1670, 1697, 3749, 3848, 4488,
    4553, 4574
\__enumext_mini_right_cmd:n 60, 1655, 1657, 1657
\__enumext_mini_set_vskip_vii: 58, 1564, 1564,
    1623
\__enumext_mini_set_vskip_viii: 58, 1564, 1586,
    1629
\__enumext_minipage:w 36, 377, 385, 573, 4511, 4813,
    5064
\l__enumext_minipage_active_v_bool 3845, 3868,
    3893
\g__enumext_minipage_active_vii_bool . 116,
    4500, 4509, 4531
\l__enumext_minipage_active_vii_bool . 4482,
    4493
\g__enumext_minipage_active_viii_bool 4564,
    4572, 4591
\l__enumext_minipage_active_viii_bool 4547,
    4558
\g__enumext_minipage_active_X_bool . . . 177
\l__enumext_minipage_active_X_bool . . . . 87
\__enumext_minipage_add_space: . 55, 102, 1313,
    1339, 3747
\g__enumext_minipage_after_skip 87, 1568, 1580,
    4529, 4589
\l__enumext_minipage_after_skip . . 54, 102, 87,
    1326, 1366, 1368, 1373, 1376, 1380, 1385, 1389, 1392,
    1396, 1408, 1413, 1416, 1420, 1425, 1429, 1432, 1436,
    1447, 1452, 1455, 1459, 1464, 1468, 1471, 1475, 1487,
    1501, 1534, 1536, 1541, 1543, 1545, 1549, 1553, 1555,
    1557, 1588, 1601, 1615, 1666, 1693, 3903
\g__enumext_minipage_center_vii_bool . 4515,
    4532
\g__enumext_minipage_center_viii_bool 4576,
    4592
\g__enumext_minipage_center_X_bool . . . 177
\l__enumext_minipage_hsep_v_dim . . . . . 3843
\l__enumext_minipage_hsep_vii_dim . . . . 4480
\l__enumext_minipage_hsep_viii_dim . . . 4545
\l__enumext_minipage_left_skip 87, 1488, 1566,
    1571, 1575, 1589, 1593, 1607, 1625, 1631
\l__enumext_minipage_left_v_dim . . 3841, 3848
\l__enumext_minipage_left_vii_dim 4476, 4488
\l__enumext_minipage_left_viii_dim 4541, 4553
\l__enumext_minipage_left_X_dim . . . . . 87
\g__enumext_minipage_right_skip 87, 1567, 1572,
    1576, 4514, 4575
\l__enumext_minipage_right_skip . 54, 87, 1315,
    1321, 1326, 1328, 1330, 1489, 1490, 1496, 1501, 1502,
    1503, 1508, 1590, 1597, 1611, 1672, 1699
\l__enumext_minipage_right_v_dim . 1688, 1697,
    3839, 3843
\g__enumext_minipage_right_vii_dim 115, 4484,
    4511, 4534
\l__enumext_minipage_right_vii_dim 115, 4474,
    4479, 4485
\g__enumext_minipage_right_viii_dim . . 4549,
    4574, 4594
\l__enumext_minipage_right_viii_dim . . 4539,
    4544, 4550
\g__enumext_minipage_right_X_dim . . . . . 177
\g__enumext_minipage_right_X_skip . . . . 177
\__enumext_minipage_set_skip: . 54, 1313, 1313,
    1341
\g__enumext_minipage_stat_int . . 102, 87, 1677,
    1704, 3746, 3757, 3762, 3846, 3895, 3900
\l__enumext_minipage_temp_skip 87, 1387, 1397,
    1400, 1427, 1437, 1440, 1466, 1476, 1479, 1551, 1558,
    1560
\l__enumext_miniright_code_vii_box 4522, 4526
\g__enumext_miniright_code_vii_tl 116, 4517,
    4524, 4533
\l__enumext_miniright_code_viii_box . . 4583,
    4587
\g__enumext_miniright_code_viii_tl 4578, 4585,
    4593
\l__enumext_miniright_code_X_box . . . . . 177
\l__enumext_mode_box_bool . . . . 638, 3303, 3473
\__enumext_multi_addvspace: 53, 101, 1268, 1268,
    3708
\__enumext_multi_set_vskip: 52, 1223, 1223, 1270
\l__enumext_multicols_above_ii_skip . . 1242
\l__enumext_multicols_above_iii_skip . . 1251
\l__enumext_multicols_above_iv_skip . . 1260
\l__enumext_multicols_above_v_skip 1287, 1301,
    1311, 1502
\l__enumext_multicols_above_X_skip . . . . 79
\l__enumext_multicols_below_ii_skip . . 1369,
    1378, 1382, 1394, 1399
\l__enumext_multicols_below_iii_skip . 1409,
    1418, 1422, 1434, 1439
\l__enumext_multicols_below_iv_skip . . 1448,
    1457, 1461, 1473, 1478
\l__enumext_multicols_below_v_skip 1291, 1305,
    1503, 1537, 1544, 1546, 1556, 1559, 3885

```

```

\l__enumext_multicols_below_X_skip . . . . 79
\g__enumext_multicols_right_X_skip . . . . 79
\__enumext_multicols_start: 101, 102, 3684, 3684,
3751
\__enumext_multicols_stop: 101, 1662, 3714, 3714,
3767
\__enumext_nested_base_line_fix: 47, 100, 975,
981, 3639
\__enumext_newlabel:nn 31, 36, 79, 423, 423, 2581,
3099
\l__enumext_newlabel_arg_one_tl 31, 36, 78, 88,
163, 2574, 2582, 2644, 3088, 3100, 3138
\l__enumext_newlabel_arg_two_tl 31, 36, 78, 163,
2530, 2540, 2553, 2568, 2583, 3075, 3080, 3085, 3101
\__enumext_parse_foreach_keys:n . . 5363, 5379,
5396
\__enumext_parse_foreach_keys:nn . 5363, 5386,
5398
\__enumext_parse_keys:n 47, 64, 3630, 3630, 3792
\__enumext_parse_keys_vii:n 64, 4599, 4637, 4637
\__enumext_parse_keys_viii:n . 4862, 4907, 4907
\__enumext_parse_save_key:n 75, 2374, 2379, 2379
\__enumext_parse_save_key_vii:n 75, 2369, 2379,
2387
\__enumext_parse_series:n . . 64, 100, 118, 1862,
1862, 3638, 4643
\__enumext_parse_store_keys:n . . . . . 100
\l__enumext_parsep_i_skip . . . . . 1240, 1244
\l__enumext_parsep_ii_skip . . . . . 1249, 1253
\l__enumext_parsep_iii_skip . . . . . 1258, 1262
\l__enumext_parsep_vii_skip . . . . . 4815
\l__enumext_parsep_viii_skip . . . . . 5066
\l__enumext_partopsep_v_skip . 1303, 1307, 1498,
1521
\l__enumext_partopsep_viii_skip . . . . . 1599
\__enumext_phantomsection: 36, 388, 416, 420, 436
\__enumext_pre_itemsep_skip: . . 55, 1331, 1360,
1360
\__enumext_print_footnote: . . 438, 459, 522, 527
\__enumext_print_footnote_mini: 488, 549, 554
\__enumext_print_footnote_standar: 500, 516,
580
\__enumext_print_footnote_starred: 500, 545,
560, 564
\__enumext_print_keyans_box:NN 77, 2498, 2498,
2511, 2655, 2659, 3185, 3225, 5008, 5023
\l__enumext_print_keyans_i_tl . . . . 5147, 5169
\l__enumext_print_keyans_ii_tl . . . 5151, 5170
\l__enumext_print_keyans_iii_tl . . 5155, 5171
\l__enumext_print_keyans_iv_tl . . . 5159, 5172
\l__enumext_print_keyans_star_bool . 47, 130,
130, 987, 995, 5192, 5197
\l__enumext_print_keyans_starred_tl 129, 130,
130, 5143, 5190
\l__enumext_print_keyans_vii_tl 129, 5163, 5173
\l__enumext_print_keyans_X_tl . . . . . 130
\__enumext_printkeyans:nnn 129, 130, 5166, 5174,
5177
\__enumext_redefine_item: . 92, 3285, 3285, 3571
\l__enumext_ref_key_arg_tl 43, 50, 232, 739, 740,
752, 783, 786, 796, 802, 812, 853, 854, 864
\l__enumext_ref_the_count_tl . 43, 50, 746, 749,
752, 791, 793, 796, 807, 809, 812, 859, 861, 864
\__enumext_regex_counter_style: . . 32, 43, 227,
227, 747, 792, 808, 860
\__enumext_register_counter_style:Nn . . 602,
602, 607, 608, 609, 610, 611
\__enumext_remove_extra_parsep_vii: . . 4617,
4839, 4839
\__enumext_remove_extra_parsep_viii: . 4879,
5094, 5094
\__enumext_renew_footnote: . . 438, 442, 506, 511
\__enumext_renew_footnote_mini: 471, 536, 541
\__enumext_renew_footnote_standar: 500, 500,
572
\__enumext_renew_footnote_starred: 500, 532,
4809, 5060
\l__enumext_renew_the_count_v_tl 862, 870, 872
\l__enumext_renew_the_count_vii_tl 794, 821,
823
\l__enumext_renew_the_count_viii_tl 810, 828,
830
\l__enumext_renew_the_count_X_tl . . . . . 50
\__enumext_rescan_anskey_env:n . . 85, 86, 2827,
2893, 2988, 2996
\__enumext_reset_global_bool: . . 338, 341, 350
\__enumext_reset_global_int: . . . 338, 340, 344
\__enumext_reset_global_tl: . . . . 338, 342, 356
\__enumext_reset_global_vars: . 34, 87, 338, 338,
3026
\l__enumext_resume_active_bool 64, 66, 61, 1866,
1986
\__enumext_resume_counter: . . 66, 67, 1984, 1990,
1997
\__enumext_resume_counter:n . 64, 66, 1955, 1960,
1984, 1984, 2054, 2062
\__enumext_resume_counter_save_ans: 67, 1984,
1995, 2027
\__enumext_resume_counter_series: 66, 67, 1984,
1993, 2010
\g__enumext_resume_int . . . 61, 1907, 2001, 2002
\__enumext_resume_last:n 64, 65, 1862, 1868, 1881
\l__enumext_resume_name_tl 61, 1903, 1911, 1914,
1930, 1938, 1941, 1987, 1988, 2016, 2023
\__enumext_resume_save_counter: . 65, 102, 119,
1894, 1894, 3773, 4661
\__enumext_resume_series:n . 66, 1830, 1951, 1951
\__enumext_resume_starred: . 68, 1831, 2048, 2048
\g__enumext_resume_vii_int 61, 1934, 2006, 2007
\l__enumext_rightmargin_vii_dim . . 4330, 4334,
4339
\l__enumext_rightmargin_viii_dim . 4361, 4365,
4370
\__enumext_safe_exec: . . 39, 99, 3619, 3619, 3791
\__enumext_safe_exec_vii: . 39, 4598, 4620, 4620
\__enumext_safe_exec_viii: 124, 4861, 4883, 4883
\__enumext_second_part: . . 102, 3753, 3753, 3805
\__enumext_second_part_v: . . . 3835, 3891, 3939
\l__enumext_series_name_tl . . . . . 66
\l__enumext_series_str . 65, 100, 118, 1828, 1864,
1872, 1873, 1875, 1877, 1898, 1901, 1905, 1925, 1928,
1932, 3634, 4641
\__enumext_set_error:nn . . . . . 5273, 5310, 5312
\__enumext_set_item_width: 103, 3775, 3775, 3801
\__enumext_set_parse:n . . . . . 5273, 5284, 5300
\l__enumext_setkey_tmpa_int . . . 121, 5277, 5281
\l__enumext_setkey_tmpa_seq . . 121, 5275, 5285,
5291, 5293, 5295, 5307

```

```

\l__enumext_setkey_tmpa_tl . . . 121, 5283, 5287
\l__enumext_setkey_tmpb_seq . . 121, 5276, 5279,
    5283, 5284
\l__enumext_setkey_tmpb_tl 121, 5302, 5304, 5305
\l__enumext_show_answer_bool . 2314, 2338, 2667,
    3155, 3169, 4220, 5006
\__enumext_show_length:nnn . . 49, 235, 235, 5521,
    5522, 5523, 5524, 5525, 5526, 5527, 5528, 5529, 5530,
    5536, 5537, 5538, 5539, 5540, 5541, 5542, 5543, 5544,
    5545
\l__enumext_show_position_bool . . . 2317, 2341,
    2671, 3159, 3170, 4221, 5010
\g__enumext_standar_bool 33, 99, 34, 257, 260, 279,
    353, 502, 518, 1896, 1961, 1973, 1999, 2012, 2050,
    2190, 2204, 2535, 2548, 2563, 3654
\l__enumext_standar_bool 99, 102, 34, 2536, 3626,
    3772, 4634
\l__enumext_standar_first_bool 33, 99, 34, 284,
    1883, 2030, 2092, 2099
\__enumext_standar_item_vii:w . 120, 4691, 4709,
    4711
\__enumext_standar_item_viii:w 125, 126, 4934,
    4952, 4954
\__enumext_standar_ref: . . . . 43, 737, 756, 3573
\__enumext_standar_ref:n . . . . 43, 729, 737, 737
\g__enumext_standar_series_tl . 61, 1885, 1886,
    2052, 2055
\__enumext_standar_unknown_keys:n 3403, 3407,
    3411
\__enumext_standar_unknown_keys:nn 3403, 3413,
    3415
\g__enumext_starred_bool 33, 118, 34, 267, 270, 294,
    354, 1923, 1966, 1977, 2004, 2019, 2058, 2164, 2210,
    2526, 3069, 4535
\l__enumext_starred_bool 118, 119, 124, 34, 1647,
    2564, 2599, 2605, 2653, 2942, 2947, 3178, 3191, 3627,
    4633, 4660, 4895, 4899
\__enumext_starred_columns_set_vii: . . 4312,
    4312, 4608
\__enumext_starred_columns_set_viii: . 4312,
    4343, 4870
\l__enumext_starred_first_bool 33, 118, 34, 299,
    985, 994, 1888, 2039, 2092, 2099
\__enumext_starred_item:nn . . . 3248, 3248, 3291
\__enumext_starred_item_exec: . 126, 4968, 4998,
    5038
\__enumext_starred_item_vii:w . 120, 121, 4691,
    4708, 4725
\__enumext_starred_item_vii_aux_i:w . . 4691,
    4730, 4733
\__enumext_starred_item_vii_aux_ii:w . 4691,
    4731, 4736, 4738
\__enumext_starred_item_vii_aux_iii:w 4691,
    4741, 4748
\__enumext_starred_item_viii:w 125, 126, 4951,
    4968, 4968
\__enumext_starred_item_viii_aux_i:w . . 126,
    4968, 4973, 4976
\__enumext_starred_item_viii_aux_ii:w . 126,
    4968, 4974, 4991, 4993
\__enumext_starred_joined_item_vii:n 114, 120,
    4374, 4374, 4706
\__enumext_starred_joined_item_viii:n . 114,
    125, 4374, 4423, 4949
\__enumext_starred_ref: . . . . 44, 781, 817, 3604
\__enumext_starred_ref:n . . . . 43, 775, 781, 781
\g__enumext_starred_series_tl . 61, 1890, 1891,
    2060, 2063
\__enumext_starred_unknown_keys:n 3385, 3387,
    3389
\__enumext_starred_unknown_keys:nn 3385, 3391,
    3393
\__enumext_start_from:NNn 45, 875, 875, 888, 910,
    916
\l__enumext_start_i_int . . . . . 2002, 2014, 2033
\__enumext_start_item_tmp_vii: 118, 4611, 4691,
    4691
\__enumext_start_item_tmp_viii: . . 4873, 4934,
    4934
\__enumext_start_item_vii:w 120, 122, 4717, 4722,
    4745, 4752, 4800, 4800
\__enumext_start_item_viii:w . . 126, 4960, 4965,
    4996, 5051, 5051
\g__enumext_start_line_tl 33, 34, 287, 302, 359,
    2234, 2239, 2244, 2258, 2263, 2268
\__enumext_start_list:nn . 35, 97, 377, 379, 3795,
    3928, 4602, 4864
\__enumext_start_list_tag:n . . 3941, 3965, 4810,
    5061
\__enumext_start_mini_vii: 119, 4472, 4472, 4652
\__enumext_start_mini_viii: . . 124, 4537, 4537,
    4918
\__enumext_start_save_ans_msg: . . 68, 69, 2076,
    2076, 2101
\__enumext_start_store_level: . 100, 3648, 3648,
    3794
\__enumext_start_store_level_vii: 119, 4601,
    4663, 4663
\l__enumext_start_vii_int . . . 2007, 2021, 2042
\l__enumext_start_X_int . . . . . 100
\__enumext_stop_item_tmp_vii: . . 118, 120, 122,
    4610, 4616, 4693, 4802
\__enumext_stop_item_tmp_viii: 125, 4872, 4878,
    4936, 5053
\__enumext_stop_item_vii: 122, 4800, 4802, 4822
\__enumext_stop_item_viii: . . . 5051, 5053, 5077
\__enumext_stop_list: 35, 116, 119, 377, 380, 3719,
    3727, 3881, 3888, 4495, 4503, 4560, 4567
\__enumext_stop_list_tag:n . . . 3941, 3981, 4825,
    5080
\__enumext_stop_mini_vii: 116, 119, 4472, 4491,
    4656
\__enumext_stop_mini_viii: 125, 4537, 4556, 4922
\__enumext_stop_save_ans_msg: . 68, 2076, 2081,
    3015
\__enumext_stop_start_list_tag: . . 3941, 3973,
    4812, 5063
\__enumext_stop_store_level: . . 101, 3677, 3677,
    3720, 3728
\__enumext_stop_store_level_vii: . . 116, 119,
    4496, 4504, 4663, 4673
\l__enumext_store_active_bool 30, 69, 112, 2031,
    2040, 2108, 2740, 3652, 3665, 3813, 3821, 4173, 4665,
    4675, 4885, 4901
\__enumext_store_active_keys:n . 74, 100, 2347,
    2347, 3645
\__enumext_store_active_keys_vii:n . 74, 118,
    2347, 2357, 4644
\__enumext_store_addto_prop:n 76, 87, 2422, 2422,
    2430, 2590, 3050, 5001

```



`\__enumext_store_addto_seq:n` 76, 89, 2431, 2431, 2435, 2442, 2456, 2464, 2473, 2487, 2495, 2648, 3143  
`\l__enumext_store_anskey_arg_tl` . . 30, 79, 112, 2596, 2601, 2603, 2608, 2615, 2618, 2628, 2633, 2636, 2642, 2648  
`\__enumext_store_anskey_code:n` 79, 81, 86, 2587, 2587, 2733, 2986, 2994  
`\l__enumext_store_anskey_env_tl` . . 30, 85, 112, 2916, 2920, 2926, 2988, 2996  
`\l__enumext_store_anskey_opt_tl` . . 30, 86, 112, 2917, 2944, 2950, 2957, 2963, 2973, 2983, 2992  
`\__enumext_store_anskey_safe_outer:` . . . . 82  
`\g__enumext_store_columns_break_bool` . 2840, 2941, 3003  
`\l__enumext_store_columns_break_bool` . 2598, 2689  
`\l__enumext_store_current_label_tl` 30, 87, 89, 126, 112, 3031, 3034, 3037, 3043, 3048, 3050, 3107, 3110, 3113, 3119, 3124, 3134, 3143, 4978, 4983, 4987, 5000, 5001, 5003  
`\l__enumext_store_current_label_tmp_tl` . 30, 112, 3439, 3444  
`\l__enumext_store_current_opt_arg_tl` 30, 126, 112, 3153, 3166, 3172, 4989  
`\__enumext_store_internal_ref:` . . 77, 79, 2512, 2512, 2593  
`\g__enumext_store_item_join_int` . . 2843, 2948, 2952, 3004  
`\l__enumext_store_item_join_int` . . 2606, 2610, 2692  
`\g__enumext_store_item_star_bool` . 2845, 2955, 3005  
`\l__enumext_store_item_star_bool` . 2613, 2694  
`\g__enumext_store_item_symbol_sep_dim` 2850, 2970, 2975, 3007  
`\l__enumext_store_item_symbol_sep_dim` 2625, 2630, 2699  
`\g__enumext_store_item_symbol_tl` . 2848, 2961, 2965, 3006  
`\l__enumext_store_item_symbol_tl` . 2616, 2620, 2697  
`\l__enumext_store_keyans_item_opt_sep_-tl` . . . . 2300, 3041, 3045, 3117, 3121, 4981, 4985  
`\__enumext_store_level_close:` 76, 2436, 2460, 3681  
`\__enumext_store_level_close_vii:` . 77, 2467, 2491, 4679  
`\__enumext_store_level_open:` 76, 100, 2436, 2436, 3660, 3673  
`\__enumext_store_level_open_vii:` . . 77, 2467, 2467, 4669  
`\g__enumext_store_name_tl` 30, 69, 112, 358, 365, 366, 367, 368, 2084, 2110, 2233, 2238, 2243, 2257, 2262, 2267, 3013  
`\l__enumext_store_name_tl` 30, 69, 70, 112, 1917, 1920, 1944, 1947, 2035, 2044, 2079, 2088, 2089, 2110, 2111, 2112, 2114, 2115, 2117, 2119, 2120, 2122, 2124, 2125, 2149, 2424, 2426, 2433, 2576, 2577, 2679, 2922, 3090, 3091, 3204, 3217, 5018  
`\l__enumext_store_ref_key_bool` 79, 2323, 2591, 2639, 3054, 3131  
`\l__enumext_store_save_key_vii_bool` . . 2359, 2389  
`\l__enumext_store_save_key_vii_tl` 2361, 2362, 2390, 2391, 2471, 2479, 2483, 2487  
`\l__enumext_store_save_key_X_bool` . . 74, 130  
`\l__enumext_store_save_key_X_tl` . . . . 74, 130  
`\l__enumext_store_upper_level_X_bool` . . 130  
`\__enumext_storing_exec:` 69, 84, 2086, 2102, 2106  
`\__enumext_storing_set:n` 68, 69, 2071, 2086, 2086  
`\l__enumext_the_counter_v_tl` . . . . . 861  
`\l__enumext_the_counter_vii_tl` . . . . . 793  
`\l__enumext_the_counter_viii_tl` . . . . . 809  
`\l__enumext_the_counter_X_tl` . . . . . 50  
`\__enumext_tmp:n` 45, 49, 54, 60, 71, 78, 79, 86, 94, 99, 100, 111, 134, 141, 166, 170, 177, 197, 634, 643, 1824, 1835, 2067, 2075, 2128, 2146, 2287, 2328, 2329, 2346, 2365, 2378, 2514, 2521, 2522, 2543, 2556, 2559, 2570, 3056, 3063, 3363, 3370, 3403, 3410, 3543, 3583, 3584, 3618  
`\__enumext_tmp:nn` 644, 665, 666, 700, 701, 716, 905, 930, 1007, 1029, 1030, 1050, 1104, 1112, 1113, 1127, 1192, 1208, 1209, 1222, 1713, 1729, 3347, 3362  
`\__enumext_tmp:nnn` 717, 733, 734, 735, 736, 763, 779, 780  
`\__enumext_tmp:nnnnnn` 931, 956, 959, 962, 964, 966, 969, 972  
`\__enumext_tmp:w` . . . . . 5122, 5125  
`\l__enumext_tmpa_vii_int` 4322, 4325, 4334, 4365  
`\l__enumext_tmpa_viii_int` . . . . . 4353, 4356  
`\l__enumext_tmpa_X_dim` . . . . . 177  
`\l__enumext_tmpa_X_int` . . . . . 177  
`\l__enumext_topsep_v_skip` 1289, 1293, 1492, 4166  
`\l__enumext_topsep_vii_skip` . . 1569, 1578, 1582  
`\l__enumext_topsep_viii_skip` . 1591, 1613, 1617  
`\__enumext_undefine_anskey_env:` . 83, 87, 2773, 2773, 3024  
`\__enumext_unskip_unkern:` . . 33, 241, 241, 1342, 1514, 3722, 3723, 3763, 3883, 3884, 3901, 4816, 4817, 5067, 5068  
`\l__enumext_vspace_a_star_v_bool` . . . . . 1762  
`\l__enumext_vspace_a_star_vii_bool` . . . 1784  
`\l__enumext_vspace_a_star_viii_bool` . . 1795  
`\l__enumext_vspace_a_star_X_bool` . . . . . 100  
`\__enumext_vspace_above:` 61, 102, 1730, 1730, 3733  
`\__enumext_vspace_above_v:` . 62, 1758, 1758, 3837  
`\l__enumext_vspace_above_v_skip` . . 1760, 1764, 1766  
`\__enumext_vspace_above_vii:` 62, 119, 1780, 1780, 4649  
`\l__enumext_vspace_above_vii_skip` 1782, 1786, 1788  
`\__enumext_vspace_above_viii:` . 62, 1780, 1791, 4916  
`\l__enumext_vspace_above_viii_skip` 1793, 1797, 1799  
`\l__enumext_vspace_b_star_v_bool` . . . . . 1773  
`\l__enumext_vspace_b_star_vii_bool` . . . 1806  
`\l__enumext_vspace_b_star_viii_bool` . . 1817  
`\l__enumext_vspace_b_star_X_bool` . . . . . 100  
`\__enumext_vspace_below:` 62, 102, 1744, 1744, 3771  
`\__enumext_vspace_below_v:` . 62, 1769, 1769, 3910  
`\l__enumext_vspace_below_v_skip` . . 1771, 1775, 1777  
`\__enumext_vspace_below_vii:` 63, 119, 1802, 1802, 4659  
`\l__enumext_vspace_below_vii_skip` 1804, 1808, 1810  
`\__enumext_vspace_below_viii:` . 63, 1802, 1813,

4924	
\l_enumext_vspace_below_viii_skip	1815, 1819, 1821
\_enumext_widest_from:nNNn	45, 889, 889, 904, 923
\g_enumext_widest_label_tl	28, 40, 67, 622, 626, 630
\l_enumext_wrap_label_opt_v_bool	3433
\l_enumext_wrap_label_opt_vii_bool	120, 4716
\l_enumext_wrap_label_opt_viii_bool	126, 4959
\l_enumext_wrap_label_opt_X_bool	100
\l_enumext_wrap_label_v_bool	3429, 3433, 3441, 3488, 3503
\l_enumext_wrap_label_vii_bool	120, 4716, 4720, 4728, 4792
\l_enumext_wrap_label_viii_bool	126, 4959, 4963, 4971, 5043
\l_enumext_wrap_label_X_bool	100
\_enumext_wrapper_label_v:n	3490, 3505, 4229
\_enumext_wrapper_label_vii:n	4794
\_enumext_wrapper_label_viii:n	5045
\l_enumext_write_aux_file_tl	31, 79, 89, 163, 2579, 2585, 3097, 3103
enumext*	5, 4596
enumXi	585
enumXii	585
enumXiii	585
enumXiv	585
enumXv	585
enumXvi	585
enumXvii	585
enumXviii	585
Environments provide by enumext:	
anskey*	30, 69, 74, 78, 80, 83-85, 87, 100, 101, 119, 128, 129, 135, 137
enumext*	27, 28, 31-33, 37-41, 43, 44, 46, 48-52, 58, 59, 62-66, 68-71, 73-79, 81-83, 86-88, 93, 94, 99, 100, 106, 113, 114, 116, 117, 119, 121-125, 127-130, 132, 136, 139, 140
enumext	27, 28, 32, 33, 38-43, 45-54, 57, 60-66, 68-71, 73-76, 78, 79, 81-83, 86-88, 91-93, 95, 97, 100, 103, 104, 108, 113, 115, 118, 119, 121, 124, 129, 130, 132, 136, 137, 139
keyans*	27, 28, 30-34, 37, 38, 40, 43-46, 48-52, 58, 59, 62, 63, 69, 70, 73, 74, 76, 83, 88, 94, 99, 106, 113, 115, 123, 124, 136, 138, 140
keyanspic	27, 28, 30, 31, 34, 40, 44, 69, 70, 73, 76, 83, 87-90, 94, 106-110, 112, 138
keyans	27, 28, 30, 31, 33, 34, 38, 40, 41, 44, 46, 48-51, 53, 57, 60-62, 69, 70, 73, 74, 76, 83, 87-90, 94-97, 103-105, 107-109, 111, 115, 125, 136, 138
Environments:	
center	113
description	113
enumerate	113
flushleft	113
flushright	113
itemize	113
list	32, 35, 81, 97, 101, 103, 106, 107, 109, 113, 116
lrbox	122
minipage	32, 35, 36, 39, 52, 54, 55, 107, 108, 110, 112, 113, 116, 122
multicols	52-55, 60, 101, 102
quotation	113

quote	113
scontents	84, 86
tabbing	113
trivlist	113
verbatim	113
verse	113
exp commands:	
\exp_after:wN	5125
\exp_args:Ne	2985, 2993, 3642, 5113
\exp_args:NV	2705, 2860, 3373, 3391, 3413, 5398
\exp_not:N	58, 625, 752, 796, 812, 864, 1060, 1063, 1074, 1075, 1076, 1087, 1088, 1099, 1100, 2644, 2676, 2677, 3136, 3201, 3202, 3214, 3215, 5015, 5016, 5122
\exp_not:n	289, 304, 317, 325, 333, 691, 711, 752, 796, 812, 864, 1061, 1851, 1860, 2311, 2408, 2420, 2582, 2610, 2620, 2630, 2644, 2645, 2952, 2965, 2975, 3100, 3138, 3140, 4050, 5227, 5237, 5430, 5435
F	
\fbox	2294
\fboxrule	2294
\fboxsep	2294
file commands:	
\file_input_stop:	5834
first	1113
font	644
\footnote	37
\footnote	37, 444, 473
\footnotemark	454, 483
\footnotesize	2677, 3202, 3215, 5016
\footnotetext	440
\foreachkeyans	17, 133, 5363
G	
\getkeyans	17, 128, 5111
group commands:	
\group_begin:	2675, 2720, 2895, 2982, 3200, 3213, 5014, 5168
\group_end:	2682, 2736, 2999, 3207, 3220, 5021, 5175
H	
\hbadness	4827, 5082
hbox commands:	
\hbox_overlap_left:n	3281, 4785
\hbox_set:Nn	614, 4096
\hbox_set_end:	4826, 5081
\hbox_set_to_wd:Nnw	4803, 5054
\hfill	674, 679, 685, 686, 1669, 1696, 2644, 3136, 4499, 4563
hook commands:	
\hook_gput_code:nnn	9, 207, 211, 215, 388
\hook_gremove_code:nn	86, 2911
\hook_gset_rule:nnnn	389
\hook_if_empty:nTF	2909
\hyperlink	80, 89
\hyperlink	2644, 3136
\hypertarget	36
\hypertarget	415
I	
\IfDocumentMetadataTF	504, 520, 534, 547, 3298, 3468, 3967, 3975, 3983, 4017, 4025, 4033, 4119, 4129, 4137, 4143, 4148, 4194, 4203, 4287, 4295, 4497, 4561, 4607, 4615, 4761, 4869, 4877
\IfHyperBoolean	396
\IfPackageLoadedTF	11, 19, 392, 405

`\ignorespaces` . . . 1063, 1076, 1088, 1100, 4612, 4689, 4722, 4745, 4752, 4798, 4818, 4874, 4932, 4965, 4996, 5049, 5069

`\inputlineno` . . . . . 289, 304, 317, 325, 333

int commands:

`\int_add:Nn` . . . . . 4407, 4456

`\int_case:nn` . . . . . 1237, 1362, 2159, 2185, 2224, 2248

`\int_case:nnTF` . . . . . 243

`\int_compare:nNnTF` . . . 568, 784, 800, 819, 826, 1332, 1351, 1505, 1523, 1635, 1651, 1663, 1691, 2272, 2278, 2744, 2748, 2752, 2760, 2806, 2810, 2814, 3011, 3032, 3073, 3078, 3083, 3108, 3196, 3624, 3635, 3657, 3670, 3686, 3701, 3716, 3757, 3822, 3826, 3854, 3879, 3895, 4068, 4177, 4181, 4377, 4387, 4403, 4426, 4436, 4452, 4625, 4629, 4667, 4677, 4829, 4841, 4890, 4902, 5084, 5096, 5281, 5413

`\int_compare_p:nNn` . . . 258, 268, 280, 281, 295, 296, 1641, 1642, 2165, 2191, 2527, 2537, 2549, 2550, 2565, 2606, 2783, 2784, 2795, 2796, 2948, 3667

`\int_decr:N` . . . . . 4406, 4455

`\int_eval:n` . . . 375, 918, 2426, 2577, 2677, 3091, 3202, 3215, 3558, 3603, 4395, 4444, 5016

`\int_from_alph:n` . . . . . 883, 897

`\int_from_roman:n` . . . . . 885, 899

`\int_gadd:Nn` . . . . . 4408, 4457

`\int_gdecr:N` . . . . . 2168, 2173, 2177, 2181, 2194

`\int_gincr:N` 2001, 2006, 2589, 3146, 3235, 3269, 3448, 3746, 3846, 4218, 4695, 4771, 4938, 5005

`\int_gset:Nn` . . . . . 452, 481, 2217

`\int_gset_eq:NN` . . . 449, 478, 1900, 1907, 1913, 1919, 1927, 1934, 1940, 1946

`\int_gzero:N` . . . 346, 347, 348, 1677, 1704, 2284, 3004, 3762, 3900, 4852, 5108

`\int_if_exist:NTF` 1875, 1911, 1917, 1938, 1944, 2122

`\int_incr:N` . . . 2759, 3623, 3817, 4067, 4624, 4694, 4889, 4937

`\int_mod:nn` . . . . . 4843, 5098

`\int_new:N` . . . 28, 29, 30, 31, 32, 33, 61, 62, 87, 104, 123, 144, 145, 156, 157, 158, 160, 171, 172, 180, 181, 182, 183, 184, 1877, 2125

`\int_set:Nn` 879, 883, 885, 2014, 2021, 2033, 2042, 2896, 4281, 4282, 4322, 4353, 4376, 4382, 4398, 4425, 4431, 4447, 4827, 5082, 5277, 5415

`\int_set_eq:NN` . . . . . 2002, 2007, 4405, 4454

`\int_sign:n` . . . . . 2219

`\int_step_function:nnN` . . . . . 2543, 2556, 2570

`\int_step_function:nnnN` . . . . . 5419

`\int_step_inline:nn` . . . . . 5329

`\int_step_inline:nnn` . . . . . 4283

`\int_to_roman:n` . . . . . 219, 2523, 2560

`\int_use:N` 368, 373, 374, 1333, 1352, 1664, 2016, 2023, 2035, 2044, 3558, 3578, 3603, 3643, 3687, 3696, 3711, 3717, 4380, 4381, 4393, 4429, 4430, 4442, 5750, 5754, 5760, 5764

`\int_zero:N` . . . . . 4833, 5088

`\item` . . . 91, 95, 120, 122, 125, 127, 381, 2444, 2450, 2475, 2481, 2603, 3110, 3113, 3287, 3452, 4123, 4125, 4609, 4611, 4871, 4873, 5003

`\item*` . . . . . 5, 15, 73, 3450

`item-pos*` . . . . . 3347

`item-sym*` . . . . . 3347

`\itemindent` . . . . . 98

`\itemindent` . . . . . 97

`itemindent` . . . . . 1007

`\itemsep` . . . . . 4112

`\itemwidth` . . . 584, 2294, 3777, 3783, 3914, 3920, 4416, 4420, 4465, 4469

K

keyans . . . . . 15, 3923

keyans\* . . . . . 15, 4859

keyanspic . . . . . 16, 4114

Keys for `\anskey` provide by `enumext`:

`break-col` . . . . . 79, 81, 84, 86

`item-join` . . . . . 79, 81, 84, 86

`item-pos*` . . . . . 79, 81, 84, 86

`item-star` . . . . . 79, 81, 84, 86

`item-sym*` . . . . . 79, 81, 84, 86

Keys for `anskey*` provide by `enumext`:

`break-col` . . . . . 79, 81, 84, 86

`item-join` . . . . . 79, 81, 84, 86

`item-pos*` . . . . . 79, 81, 84, 86

`item-star` . . . . . 79, 81, 84, 86

`item-sym*` . . . . . 79, 81, 84, 86

Keys for environments provide by `enumext`:

`above*` . . . . . 29, 48, 61, 62, 102, 119

`above` . . . . . 29, 48, 61, 62, 102, 119, 124

`after` . . . . . 50, 102, 119, 125

`align` . . . . . 29, 41, 91, 93, 96, 121, 135

`base-fix` . . . . . 47, 64, 75, 100

`before*` . . . . . 50, 102, 119, 124

`before` . . . . . 50

`below*` . . . . . 29, 61–63, 102, 119

`below` . . . . . 29, 61–63, 102, 119, 125

`check-ans` . . . 31–33, 68–73, 75, 87, 89, 102, 103, 119, 123, 136

`columns-sep` . . . . . 51, 101, 123

`columns` . . . . . 29, 51, 61, 101

`first` . . . . . 50, 122

`font` . . . . . 41, 93, 96, 111, 121

`item-pos*` . . . . . 92, 93

`item-sym*` . . . . . 30, 92, 93

`itemindent` . . . . . 29, 48, 49, 91, 92, 95, 96, 122

`itemsep` . . . . . 46, 99, 123

`label-pos` . . . . . 108, 109, 111

`label-sep` . . . . . 108

`labelsep` . . . . . 41, 98, 121

`labelwidth` . . . . . 40–45, 98, 121

`label` . . . . . 28, 40, 42, 45, 109, 113

`layout-sep` . . . . . 108

`layout-sty` . . . . . 108, 110, 112

`layout-top` . . . . . 108

`lisparindent` . . . . . 99

`list-indent` . . . . . 29, 48, 109

`list-offset` . . . . . 48, 103, 105

`listparindent` . . . . . 48, 122

`mark-ans` . . . . . 73, 75, 80

`mark-pos` . . . . . 73, 74, 135

`mark-ref` . . . . . 73, 75, 77, 80

`mini-env` 29, 37–39, 51, 60, 61, 75, 102, 113, 115, 117, 119, 124

`mini-right*` . . . . . 29, 32, 52, 75, 116, 117, 119

`mini-right` . . . . . 29, 32, 52, 59, 75, 116, 117, 119

`mini-sep` . . . . . 29, 51, 75, 102

`mode-box` . . . . . 41, 91, 93, 96

`no-store` . . . . . 31, 68–70, 75, 81, 91, 92

`noitemsep` . . . . . 46

`nosep` . . . . . 46

`parindent` . . . . . 99

`parsep` . . . . . 46, 99, 109, 122



partopsep	46
ref	28, 32, 42–44, 136
resume*	28, 63, 64, 68, 69, 75, 102, 119, 131
resume	28, 35, 63–69, 75, 102, 119, 131
rightmargin	48, 113
save-ans	30, 35, 64–71, 74–76, 81–84, 87, 89, 95, 103, 110, 121, 124–126, 128, 129, 131, 136
save-key	30, 64, 74, 100, 118
save-pos	75
save-ref	31, 36, 73, 75, 77, 79, 80, 88, 89, 95, 126
save-sep	73, 75, 88, 126
series	28, 63–68, 75, 100, 102, 118, 119, 131
show-ans	73–75, 77, 79, 80, 95, 111, 126
show-length	33, 49, 136
show-pos	30, 73, 74, 77, 79, 80, 90, 95, 111, 126
start*	29, 45, 46, 64
start	29, 32, 45, 46, 64
store-key	74
topsep	46, 48, 109
widest	28, 32, 45, 46
wrap-ans	39, 73, 75, 77, 80
wrap-label*	29, 41, 91, 93, 95, 96, 120, 121, 126
wrap-label	29, 41, 91–93, 95, 96, 109, 111, 120, 121, 126
wrap-opt	73, 75, 95, 111
keys commands:	
\keys_define:nn	636, 646, 668, 703, 719, 765, 834, 907, 933, 975, 1009, 1032, 1106, 1115, 1194, 1211, 1715, 1826, 2069, 2130, 2289, 2331, 2367, 2372, 2687, 2838, 2874, 3349, 3365, 3385, 3405, 4039, 5139, 5239, 5355, 5363
\keys_if_exist_p:nn	5351, 5352
\l_keys_key_str	81, 84, 2705, 2860, 3373, 3391, 3413, 5398, 5506
\keys_precompile:nnN	129, 203, 203, 5141, 5145, 5149, 5153, 5157, 5161, 5381
\keys_set:nn	660, 1001, 1217, 1720, 1725, 1963, 1968, 2055, 2063, 2725, 3637, 3642, 3833, 4057, 4060, 4078, 4642, 4911, 5243, 5248, 5249, 5250, 5251, 5254, 5259, 5260, 5261, 5262, 5263, 5264, 5265, 5297, 5407
\keys_set_known:nn	2992
keyval commands:	
\keyval_parse:NNn	1840, 2397, 5215

L

label	717, 763, 834
label-pos	4039
label-sep	4039
Labels provide by enumext:	
\Alph*	40
\Roman*	40
\alph*	40
\arabic*	32, 40
\roman*	40
\labelsep	4108
labelsep	644
\labelwidth	40
\labelwidth	4108
labelwidth	644
\lastnodetype	243
layout-sep	4039
layout-sty	4039
layout-top	4039
\leftmargin	98
\leftmargin	97, 4108

legacy commands:	
\legacy_if:nTF	4756, 4759, 5028, 5031
\legacy_if_gset_false:n	574, 4512
\legacy_if_set_false:n	4758, 5030
\legacy_if_set_true:n	4721, 4744, 4751, 4765, 4964, 4995
\linewidth	102
\linewidth	3741, 3777, 3843, 3914, 4280, 4325, 4356, 4478, 4543
\list	379
list-indent	1007
list-offset	1007
\listparindent	4110
listparindent	1007

M

\makebox	113
\makebox	2502, 2504, 3332, 3500, 4211, 4789, 5040
\makelabel	91, 93, 96, 113
\makelabel	91, 95, 3314, 3330, 3484, 3498
mark-ans	2287
mark-pos	2287, 2329
mark-ref	2287
mini-env	1192
mini-sep	1192
\minipage	385
\miniright	11, 59, 1633, 1681, 1708, 3760, 3898
mode commands:	
\mode_if_math:TF	2768, 2822
\mode_if_vertical:TF	1271, 1299, 1319, 1343, 1494, 1515
\mode_leave_vertical:	990, 997, 1060, 1074, 2500, 3279, 4783
mode-box	634
msg commands:	
\msg_error:nn	1683, 1710, 2729, 2762, 2766, 2820, 2928, 3824, 3828, 4070, 4127, 4179, 4627, 4892, 4904, 5266, 5325
\msg_error:nnn	742, 788, 804, 856, 1637, 1644, 1649, 1679, 1706, 1975, 1979, 2094, 2711, 2770, 2788, 2800, 2808, 2812, 2816, 2824, 2866, 3379, 3397, 3419, 4631, 4897, 5127, 5136, 5208, 5313, 5344, 5353, 5390, 5411
\msg_error:nnnn	2714, 2742, 2746, 2750, 2754, 2869, 3382, 3400, 3422, 3815, 4175, 4183, 4887, 5187, 5393
\msg_error:nnnnn	690, 710, 2310, 4049
\msg_fatal:nn	3625
\msg_fatal:nnn	588
\msg_info:nnn	13, 16, 21, 24, 394, 407
\msg_line_context:	5471, 5476, 5481, 5510, 5515, 5520, 5535, 5550, 5554, 5558, 5562, 5566, 5570, 5577, 5584, 5590, 5604, 5608, 5613, 5617, 5621, 5625, 5630, 5634, 5638, 5642, 5647, 5682, 5686, 5691, 5696, 5700, 5705, 5781, 5785, 5790, 5795, 5800, 5804, 5808, 5812, 5816, 5820, 5824, 5828, 5832
\msg_log:nnn	2114, 2119, 2124
\msg_log:nnnnn	372, 2257, 2262, 2267
\msg_log:nnnnnn	364
\msg_new:nnn	5438, 5442, 5446, 5450, 5455, 5468, 5473, 5478, 5483, 5492, 5500, 5504, 5508, 5513, 5518, 5533, 5548, 5552, 5556, 5560, 5564, 5568, 5572, 5581, 5587, 5593, 5597, 5601, 5606, 5611, 5615, 5619, 5623, 5628, 5632, 5636, 5640, 5645, 5680, 5684, 5689, 5694, 5698, 5703, 5779, 5783, 5788, 5793, 5798, 5802, 5806, 5810, 5814, 5818, 5822, 5826, 5830

<code>\msg_new:nnnn</code> .. 5459, 5650, 5659, 5668, 5674, 5707, 5717, 5727, 5737, 5747, 5757, 5767, 5773	<code>\prg_replicate:nn</code> ..... 238
<code>\msg_term:nnnn</code> . 2078, 2083, 3567, 3577, 3609, 3614	<code>\prg_return_false:</code> ..... 225
<code>\msg_term:nnnnn</code> ..... 2238	<code>\prg_return_true:</code> ..... 224
<code>\msg_warning:nn</code> ..... 3759, 3897	<code>\printkeyans</code> ..... 18, 129, 5166
<code>\msg_warning:nnnn</code> 2275, 2281, 3515, 3520, 4379, 4392, 4428, 4441	prop commands:
<code>\msg_warning:nnnnn</code> ..... 2233, 2243	<code>\prop_const_from_keyval:Nn</code> ..... 5314
<code>\multicolsep</code> ..... 101	<code>\prop_count:N</code> 366, 2426, 2577, 2679, 3091, 3204, 3217, 5018, 5416
<code>\multicolsep</code> ..... 1336, 1508, 3707, 3870	<code>\prop_get:NnNTF</code> ..... 5340
<b>N</b>	<code>\prop_gput_if_not_in:Nnn</code> ..... 2424
<code>\NeedsTeXFormat</code> ..... 3	<code>\prop_if_exist:NTF</code> ..... 2112, 5131, 5409
<code>\NewCommandCopy</code> ..... 381	<code>\prop_item:Nn</code> ..... 5133, 5433
<code>\newcounter</code> ..... 591	<code>\prop_new:N</code> ..... 2115
<code>\NewDocumentCommand</code> 1633, 2717, 4171, 5111, 5166, 5273, 5322, 5400	<code>\ProvidesExplPackage</code> ..... 4
<code>\NewDocumentEnvironment</code> . 3789, 3923, 4114, 4596, 4859	<b>R</b>
<code>\newenvsc</code> ..... 2831	<code>\raggedcolumns</code> ..... 3710, 3873
<code>\newlabel</code> ..... 37	<code>\raisebox</code> ..... 4242
<code>\newlabel</code> ..... 427	<code>\ref</code> ..... 77, 88
<code>no-store</code> ..... 2128	<code>ref</code> ..... 717, 763, 834
<code>\noindent</code> ..... 3748, 4487, 4552, 4832, 5087	<code>\refstepcounter</code> ..... 4768, 5033
<code>\nointerlineskip</code> 1345, 1348, 1517, 1520, 1671, 1698, 4487, 4552	regex commands:
<code>noitemsep</code> ..... 931	<code>\regex_match:nnTF</code> .. 223, 882, 884, 896, 898, 2924
<code>\nopagebreak</code> 1282, 1310, 1345, 1348, 1517, 1520, 1624, 1630	<code>\regex_replace_once:nnN</code> ..... 231
<code>\normalfont</code> ..... 2676, 3201, 3214, 5015	<code>\renewcommand</code> ..... 752, 796, 812, 864
<code>nosep</code> ..... 931	<code>\RenewDocumentCommand</code> . 444, 473, 1681, 1708, 3287, 3314, 3330, 3452, 3484, 3498, 4125
<b>P</b>	<code>\RequirePackage</code> ..... 17, 25
Packages:	<code>resume</code> ..... 1824
<code>caption</code> ..... 116	<code>resume*</code> ..... 1824
<code>enumext</code> ..... 27, 39, 42, 68, 97, 107, 135	<code>rightmargin</code> ..... 1007
<code>enumitem</code> ..... 39, 40	<code>\Roman</code> ..... 40, 45
<code>expl3</code> ..... 113	<code>\Roman</code> ..... 610
<code>footnotehyper</code> ..... 36, 38	<code>\roman</code> ..... 40, 45
<code>hyperref</code> ..... 31, 32, 36, 37, 80, 89, 121, 134	<code>\roman</code> ..... 611, 735, 5156
<code>ltxcmd</code> ..... 35	<b>S</b>
<code>lua-visual-debug</code> ..... 54	<code>\s</code> ..... 2925
<code>multicol</code> ..... 27, 134	<code>save-ans</code> ..... 2067
<code>scontents</code> ..... 27, 82, 84	<code>save-key</code> ..... 2365
<code>shortlst</code> ..... 113, 118, 122	<code>save-ref</code> ..... 2287
<code>\par</code> .. 1282, 1310, 1348, 1520, 1624, 1630, 1666, 1671, 1693, 1698, 2652, 3724, 3885, 3903, 4157, 4160, 4300, 4514, 4529, 4575, 4589, 4832, 5087	<code>save-sep</code> ..... 2287
para commands:	scan commands:
<code>\para_end:</code> ..... 4849, 5105	<code>\scan_stop:</code> ..... 4123, 4609, 4871, 5122, 5125
<code>\parbox</code> ..... 2294	scontents internal commands:
<code>\parindent</code> ..... 4814, 5065	<code>\l_scontents_fname_out_tl</code> ..... 2884
<code>\parsep</code> ..... 52, 108	<code>\__scontents_parse_environment_keys:n</code> . 2890
<code>\parsep</code> ..... 991, 3600, 4092, 4101	<code>\__scontents_rescan_tokens:n</code> ..... 2897
<code>parsep</code> ..... 931	<code>\l_scontents_storing_bool</code> ..... 2882
<code>\parskip</code> ..... 4815, 5066	<code>\l_scontents_writing_bool</code> ..... 2883
<code>\partopsep</code> ..... 3601, 3901, 4111	seq commands:
<code>partopsep</code> ..... 931	<code>\seq_clear:N</code> ..... 5275, 5418
peek commands:	<code>\seq_const_from_clist:Nn</code> ..... 5268
<code>\peek_meaning:NTF</code> 4700, 4714, 4729, 4740, 4943, 4957, 4972	<code>\seq_count:N</code> ..... 367, 4306, 5279
<code>\peek_meaning_remove:NTF</code> ..... 4707, 4950	<code>\seq_gclear:N</code> ..... 468, 469, 497, 498
<code>\peek_remove_spaces:n</code> ..... 3456	<code>\seq_gput_right:Nn</code> ..... 455, 456, 484, 485, 2433
<code>\phantomsection</code> ..... 36	<code>\seq_if_empty:NTF</code> ..... 461, 490, 5181, 5293
<code>\phantomsection</code> ..... 416	<code>\seq_if_exist:NTF</code> ..... 2117, 5179
prg commands:	<code>\seq_if_in:NnTF</code> ..... 5185
<code>\prg_do_nothing:</code> ..... 420	<code>\seq_item:Nn</code> ..... 2922, 4293
<code>\prg_new_protected_conditional:Npnn</code> ... 221	<code>\seq_map_function:NN</code> ..... 5284
	<code>\seq_map_inline:Nn</code> ..... 5194, 5202, 5294, 5295
	<code>\seq_map_pairwise_function:NNN</code> ... 463, 492
	<code>\seq_new:N</code> 124, 125, 127, 142, 173, 174, 175, 176, 2120
	<code>\seq_pop_left:NN</code> ..... 5283

\seq_put_right:Nn	4185, 5291, 5307, 5428
\seq_set_from_clist:Nn	5276
\seq_set_map_e:NNn	5285
\seq_use:Nn	203, 204, 5424
series	1824
\setcounter	893, 897, 899, 3558, 3603, 4154
\setenumext	6, 130, 5273
\setenumextmeta	6, 132, 5314
show-ans	2287, 2329
show-length	1104
show-pos	2329
skip commands:	
\skip_add:Nn	1242, 1251, 1260, 1273, 1277, 1301, 1305, 1321, 1379, 1381, 1395, 1398, 1419, 1421, 1435, 1438, 1458, 1460, 1474, 1477, 1496, 1545, 1546, 1557, 1559, 4101, 4109
\skip_gset:Nn	1572, 1576, 1580
\skip_gzero_new:N	1567, 1568
\skip_horizontal:N	1075, 1087, 1099, 4786, 4798, 4836, 5049, 5091
\skip_horizontal:n	1061, 2501, 2509, 3280, 3282, 4685, 4784, 4818, 4928, 5069
\skip_if_eq:nnTF	1240, 1249, 1258, 1365, 1405, 1445, 1533, 1569, 1591, 1732, 1746, 1760, 1771, 1782, 1793, 1804, 1815
\skip_new:N	81, 82, 83, 88, 89, 90, 91, 92, 93, 148, 195
\skip_set:Nn	1225, 1229, 1287, 1291, 1315, 1368, 1369, 1387, 1408, 1409, 1427, 1447, 1448, 1466, 1490, 1536, 1537, 1551, 1571, 1575, 1593, 1597, 1601, 1607, 1611, 1615, 4085
\skip_set_eq:NN	1326, 1327, 1329, 1336, 1501, 1502, 1503, 1508, 3556, 3599, 3600, 4815, 5066
\skip_sub:Nn	1375, 1377, 1391, 1393, 1415, 1417, 1431, 1433, 1454, 1456, 1470, 1472, 1543, 1544, 1555, 1556
\skip_use:N	1227, 1231, 1275, 1279, 1283, 1303, 1307, 1317, 1323, 1733, 1737, 1740, 1747, 1751, 1754, 3724
\skip_vertical:N	575, 578, 999, 4513, 4527, 4851, 5107
\skip_vertical:n	998, 4850, 5106
\skip_zero:N	1335, 1349, 1487, 1488, 1489, 1507, 1521, 3601, 3707, 3870, 4111, 4112
\skip_zero_new:N	1566, 1588, 1589, 1590
\c_zero_skip	575, 578, 999, 1240, 1249, 1258, 1406, 1445, 1569, 1591, 1733, 1747, 1760, 1771, 1782, 1793, 1804, 1815, 4513, 4527, 4851, 5107
\small	5144, 5148, 5152, 5156, 5160, 5164
socket commands:	
\socket_assign_plug:nn	3969, 3977, 3985, 4019, 4027, 4035
\socket_new:nn	3941, 3989
\socket_new_plug:nnn	3942, 3949, 3957, 3990, 3997, 4006
\socket_use:n	3970, 4020
\socket_use:nn	3978, 3986, 4028, 4036
start	905
start*	905
start-list-tags	3941, 3989
\stepcounter	448, 477, 4095, 4235
stop-list-tags	3941, 3989
stop-start-tags	3941, 3989
str commands:	
\c_backslash_str	2770, 5471, 5476, 5481, 5486, 5488, 5490, 5495, 5497, 5595, 5599, 5603, 5613, 5617, 5625, 5626, 5630, 5642, 5643, 5647, 5648, 5669, 5671, 5675,

5677, 5705, 5768, 5770, 5774, 5776, 5785, 5786, 5790, 5795, 5796, 5800, 5804, 5808	
\c_colon_str	2576, 3090, 5122
\c_left_brace_str	5576, 5583, 5589
\c_right_brace_str	5576, 5583, 5589
\str_case:nn	251, 310
\str_case:nnTF	1847, 1855, 2404, 2412, 5222, 5231
\str_clear:N	3634, 4641
\str_count:n	238
\str_if_empty:N	1864, 1905, 1932
\str_if_eq:nnTF	3559, 3605, 5324
\str_if_in:nnTF	5118
\str_new:N	84, 132, 147, 190
\str_set:Nn	675, 681, 687, 706, 707, 708, 2307, 2308, 2334, 2335, 4044, 4047
\str_use:N	3334
\strutbox	1354, 1357, 1368, 1369, 1380, 1382, 1397, 1400, 1408, 1409, 1420, 1422, 1437, 1440, 1447, 1448, 1459, 1461, 1476, 1479, 1525, 1528, 1536, 1537, 1545, 1546, 1558, 1560, 1571, 1572, 1575, 1582, 1595, 1603, 1609, 1617, 4104, 4109, 4157, 4165, 4248
T	
tag commands:	
\tag_mc_begin:n	3947, 3995, 4004
\tag_mc_end:	3951, 3999, 4008
\tag_resume:n	3944, 3992, 4131, 4139, 4205, 4297, 4497, 4561
\tag_struct_begin:n	3945, 3946, 3953, 3954, 3955, 3993, 3994, 4001, 4002, 4003, 4140
\tag_struct_end:	4150, 4151
\tag_struct_end:n	3952, 3959, 3960, 3961, 3962, 4000, 4009, 4010, 4011, 4012, 4615, 4877
\tag_suspend:n	3963, 4013, 4121, 4133, 4145, 4196, 4289, 4607, 4869
\tag_tool:n	4132
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@auxout	425
\@currentenv	251, 310
\protected@write	425
tex commands:	
\tex_newlinechar:D	2896
text commands:	
\text_expand:n	5114
\textasteriskcentered	2304, 3353
\textreferencemark	2321
\thepage	431
tl commands:	
\c_space_tl	3172, 5520, 5535, 5558, 5562, 5749, 5750, 5759, 5760, 5820, 5824
\tl_clear:N	673, 680, 2285, 2351, 2361, 2382, 2390, 2596, 2916, 2917, 3031, 3107, 4978
\tl_clear_new:N	620
\tl_const:Nn	50, 604
\tl_gclear:N	358, 359, 360, 1885, 1890, 3006, 3325, 3343, 4533, 4593, 4787
\tl_gclear_new:N	1872
\tl_gput_right:Nn	605
\tl_greplace_all:Nnn	626
\tl_gset:Nn	286, 287, 301, 302, 1873, 1886, 1891, 2110, 2920, 3256, 4735
\tl_gset_eq:NN	622, 3252, 4780
\tl_if_blank:nTF	2709, 2727, 2864, 3377, 3395, 3417, 4778, 5388
\tl_if_empty:N	740, 758, 786, 802, 821, 828, 854, 870, 1898, 1903, 1925, 1930, 1988, 2052, 2060, 2089,

2149, 2440, 2471, 2616, 2961, 2983, 3013, 3041, 3117,  
3166, 3277, 4304, 4981, 5305

`\tl_if_empty:nTF` ..... 1953

`\tl_if_exist:nTF` ..... 1958

`\tl_if_novalue:nTF` 446, 475, 2723, 3039, 3115, 3151,  
3231, 3250, 3258, 3427, 3632, 4076, 4639, 4909, 4979

`\tl_map_inline:Nn` ..... 229, 623

`\tl_new:N` 42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69,  
105, 106, 107, 113, 114, 115, 116, 117, 118, 119, 120,  
121, 122, 126, 128, 129, 130, 133, 136, 137, 155, 163,  
164, 165, 168, 189

`\tl_put_left::Ne` ..... 2950

`\tl_put_left:Nn` 2448, 2479, 2601, 2944, 2957, 2963,  
2973, 3183, 3223, 4517, 4578, 5000, 5003

`\tl_put_right:Nn` 621, 750, 794, 810, 862, 2452, 2483,  
2530, 2540, 2553, 2568, 2574, 2579, 2603, 2608, 2615,  
2618, 2628, 2633, 2636, 2642, 3034, 3037, 3043, 3048,  
3075, 3080, 3085, 3088, 3097, 3110, 3113, 3119, 3124,  
3134, 4983, 4987

`\tl_remove_all:Nn` ..... 5304

`\tl_remove_once:Nn` ..... 2518, 3060

`\tl_replace_all:Nnn` ..... 625, 5339

`\tl_reverse:N` ..... 2517, 2519, 3059, 3061

`\tl_set:Nn` . 58, 255, 265, 314, 315, 322, 323, 330, 331,  
590, 674, 679, 685, 686, 739, 783, 853, 1058, 1072,  
1085, 1097, 1987, 2088, 2352, 2362, 2383, 2391, 2673,  
2884, 3153, 3198, 3211, 4989, 5012, 5302, 5338, 5408

`\tl_set_eq:NN` 631, 745, 748, 791, 793, 807, 809, 859,  
861, 2516, 3058, 3071, 3439, 3444, 4223, 4225

`\tl_to_str:n` ..... 1958, 1964, 1969, 5114

`\tl_trim_spaces:n` . . . 621, 5291, 5302, 5308, 5324

`\tl_use:N` 627, 630, 760, 823, 830, 872, 1130, 1134, 1138,  
1142, 1146, 1150, 1154, 1158, 1162, 1166, 1170, 1174,  
1178, 1182, 1186, 1190, 2506, 2523, 2531, 2542, 2555,  
2560, 2571, 3239, 3245, 3273, 3316, 3317, 3324, 3336,  
3430, 3434, 3442, 3486, 3487, 3493, 3502, 3796, 3929,

4228, 4524, 4585, 4791, 4819, 4820, 5042, 5070, 5075,  
5169, 5170, 5171, 5172, 5173, 5190, 5287, 5406

token commands:

`\token_to_str:N` ..... 427

`\topsep` ..... 3901, 4109

`topsep` ..... 931

`\topskip` ..... 1335, 1507

U

`\u` ..... 232, 2925

`\unkern` ..... 246

unknown ..... 3363, 3385, 3403

`\unskip` ..... 245

use commands:

`\use:N` ..... 239, 3321, 3340, 3798

`\use:n` ..... 1838, 2395, 5120, 5213

`\use_none:nn` ..... 419, 5345

`\usecounter` ..... 3557, 3602

V

`\value` . . . . 1901, 1907, 1914, 1920, 1928, 1934, 1941, 1947

vbox commands:

`\vbox_set:Nn` ..... 4198

`\vbox_set_top:Nn` ..... 4522, 4583

`\vspace` 991, 1737, 1740, 1751, 1754, 1764, 1766, 1775, 1777,  
1786, 1788, 1797, 1799, 1808, 1810, 1819, 1821

W

widest ..... 905

`wrap-ans` ..... 2287

`wrap-label` ..... 644

`wrap-label*` ..... 644

`wrap-opt` ..... 2287

Z

`\z` ..... 2925