

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-29<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 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>1</b>	<b>6</b>	<b>The storage system</b>	<b>11</b>
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	11
1.3	User interface	3	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	12
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>multicol</code>	3	6.2.1	Keys for <code>\anskey</code>	12
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	13
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	14
<b>2</b>	<b>The environments provided</b>	<b>4</b>	6.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	6.5	The environment <code>keyanspic</code>	15
2.2	The environment <code>enumext*</code>	5	6.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	6.6	Printing stored content	16
2.3.1	Keys for <code>\item*</code>	5	6.6.1	The command <code>\getkeyans</code>	16
2.4	The command <code>\item</code> in <code>enumext*</code>	5	6.6.2	The command <code>\foreachkeyans</code>	16
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>6</b>	6.6.3	The command <code>\printkeyans</code>	16
<b>4</b>	<b>The command <code>\setenumextmeta</code></b>	<b>6</b>	<b>7</b>	<b>Full examples</b>	<b>17</b>
<b>5</b>	<b>The <code>keyval</code> system</b>	<b>6</b>	<b>8</b>	<b>The way of non-enumerated lists</b>	<b>20</b>
5.1	Keys for label and ref	6	<b>9</b>	<b>References</b>	<b>22</b>
5.2	Keys for spaces	7	<b>10</b>	<b>Change history</b>	<b>22</b>
5.2.1	Vertical spaces	7	<b>11</b>	<b>Index of Documentation</b>	<b>23</b>
5.2.2	Horizontal spaces	8	<b>12</b>	<b>Implementation</b>	<b>25</b>
5.3	Keys for add code	9	<b>13</b>	<b>Index of Implementation</b>	<b>130</b>
5.4	Keys for start, series and resume	9			
5.5	Keys for <code>multicols</code>	10			
5.6	Keys for <code>minipage</code>	10			
5.6.1	The command <code>\miniright</code>	10			
5.6.2	The key <code>mini-right</code>	10			

## 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  $\text{\LaTeX}$  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 multicols`, vertical alignment

<sup>\*</sup>This file describes a documentation for v1.0, last revised 2024-06-29.

<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 contents[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. 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>Xze 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>Xze 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 / 144

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  $\text{MiK}\text{\TeX}$ , 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

enumext.pdf

README.md

enumext.dtx

»

»

»

»

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

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.

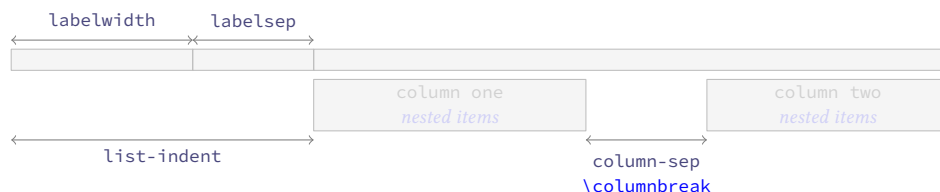


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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  $\text{\LaTeX}$  `\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.

- The implementation of `\label` and `\ref` together with the `save-ref` key are compatible with the `hyperref`[8] package.

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*}
```

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 `TeX`, `\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” 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.

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.
- ★ 3. This text is in the first level.

2.3 The command \item\*

```
\item* \item*
\item*[\langle symbol \rangle]
\item*[\langle symbol \rangle][\langle offset \rangle]
```

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 the second optional argument. The default values for `\langle symbol \rangle` and `\langle offset \rangle` are `\$star$ ‘★’` 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*` = { $\langle symbol \rangle$ } default:  $\$ \star \$$   
Sets the *symbol* to be displayed in the “left” of the box containing the current  $\langle label \rangle$  set by `labelwidth` key for `\item*` in `enumext` and `enumext*`. The *symbol* can be in text or math mode, for example `item-sym*={ $\$ \backslash ast \$$ }`.

`item-pos*` = { $\langle rigid length \rangle$ } default: *by levels*  
Sets the *offset* between the box containing the current  $\langle label \rangle$  defined by `labelwidth` key and the  $\langle symbol \rangle$  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 optional “first argument” `\item( $\langle columns \rangle$ )` 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)
8. The ninth
- Z 9. The tenth (89.28171pt)

3 The command `\setenumext`

<code>\setenumext</code>	<code>\setenumext{<math>\langle key = val \rangle</math>}</code>	<code>\setenumext[<math>\langle keyans* \rangle</math>]{<math>\langle key = val \rangle</math>}</code>
	<code>\setenumext[<math>\langle enumext, level \rangle</math>]{<math>\langle key = val \rangle</math>}</code>	<code>\setenumext[<math>\langle print, level \rangle</math>]{<math>\langle key = val \rangle</math>}</code>
	<code>\setenumext[<math>\langle enumext* \rangle</math>]{<math>\langle key = val \rangle</math>}</code>	<code>\setenumext[<math>\langle print, * \rangle</math>]{<math>\langle key = val \rangle</math>}</code>
	<code>\setenumext[<math>\langle keyans \rangle</math>]{<math>\langle key = val \rangle</math>}</code>	<code>\setenumext[<math>\langle print* \rangle</math>]{<math>\langle key = val \rangle</math>}</code>

The command `\setenumext` sets the  $\langle keys \rangle$  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  $\langle keys \rangle$  set in the optional arguments 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 {<math>\langle key name \rangle</math>}{<math>\langle key-one = val, key-two = val, ... \rangle</math>}</code>
	<code>\setenumextmeta*{<math>\langle key name \rangle</math>}{<math>\langle key-one = val, key-two = val, ... \rangle</math>}</code>
	<code>\setenumextmeta [ <math>\langle enumext* \rangle</math> ] {<math>\langle key name \rangle</math>}{<math>\langle key-one = val, key-two = val, ... \rangle</math>}</code>
	<code>\setenumextmeta [ <math>\langle enumext, level \rangle</math> ] {<math>\langle key name \rangle</math>}{<math>\langle key-one = val, key-two = val, ... \rangle</math>}</code>

The command `\setenumextmeta` adds a new “meta-key” for the environments `enumext` and `enumext*`, the  $\{ \langle key name \rangle \}$  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 version `*` will create the new “meta-key” for the environment `enumext*` and for all levels of the environment `enumext`.



## 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

`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 reference*” system with the `save-ref` key. You cannot use commands with  $\langle label \rangle$  as an argument, for example `\emph{\langle \backslash alph* \rangle}` will return an error. For full customization of how  $\langle label \rangle$  is displayed use the `font` or `wrap-label` keys.

`ref = { $\langle code \{ \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \} more code \rangle$ }` 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{\langle \backslash alph* \rangle}` 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`.

`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 = { $\langle code \{ \#1 \} more code \rangle$ }` 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* = {⟨code {#1} more code⟩}`

default: *empty*

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

## 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`.

`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.  $\TeX$  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`.

`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 “meta-key” that does not receive an argument available only for the *first level* of environment `enumext` and environment `enumext*`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` or vice versa 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  $\TeX$  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 on the “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 on the “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* 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*` = { $\langle rubber\ length \mid rigid\ length \rangle$ } default: *not used*

Set the *extra vertical 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

`itemindent` = { $\langle rigid\ length \rangle$ } default: `0pt`

Extra *horizontal indentation*, beyond `labelsep`, of the “*first line*” off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.

`rightmargin` = { $\langle rigid\ length \rangle$ } 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` = { $\langle rigid\ length \rangle$ } 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.

`list-offset` = { $\langle rigid\ length \rangle$ } 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` = { $\langle rigid\ length \rangle$ } 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 environment `enumext` the  $\langle label \rangle$  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*”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

- For the `enumext*` and `keyans*` environments the keys `list-indent` and `list-offset` have the same effect.

## 5.3 Keys for add code

- The following  $\langle keys \rangle$  should be used with “*caution*”, they are intended to inject  $\{\langle code \rangle\}$  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{\langle arg one \rangle}{\langle arg two \rangle}`. Using the `before*` key does not allow access to the `list` parameters defined by  $[\langle key = val \rangle]$ .

`before` = { $\langle code \rangle$ } default: *not used*

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

`before*` = { $\langle code \rangle$ } default: *not used*

Execute  $\{\langle code \rangle\}$  “*before*” the environment starts. The  $\{\langle code \rangle\}$  must be passed between braces, is executed “*before*” performing all calculations related to the *list parameters* and  $[\langle key = val \rangle]$  sets in the environment that is, before the arguments defining the environment are executed:  $\{\langle code \rangle\}\begin{list}{\langle arg one \rangle}{\langle arg two \rangle}$ .

`first` = { $\langle code \rangle$ } default: *not used*

Executes  $\{\langle code \rangle\}$  when “*starting*” the environment. The  $\{\langle code \rangle\}$  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}{\langle arg one \rangle}{\langle arg two \rangle}{\langle code \rangle}\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.

`after` = { $\langle code \rangle$ } default: *not used*

Execute  $\{\langle code \rangle\}$  “*after*” finishing the environment. The  $\{\langle code \rangle\}$  must be passed between braces.

## 5.4 Keys for start, series and resume

`start` = { $\langle integer \mid integer\ expression \rangle$ } default: `1`

Sets the *start value* of the numbering on the current level. The  $\{\langle integer\ expression \rangle\}$  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*` = { $\langle integer \mid string \rangle$ } default: *not used*

Sets the *start value* of the numbering on the current level. Internally  $\langle string \rangle$  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 multicols

`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.

## 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 L<sup>A</sup>T<sub>E</sub>X 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).

### 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  $\langle key \rangle$  is “*active*” the `\anskey` command and the environments `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

```
\begin{enumext}[save-ans={\store name}]
  \item Text \anskey{answer}
  \item Text
  \begin{keyans}
    ...
  \end{keyans}
\end{enumext}
```

```
\begin{enumext}[save-ans={\store name}]
  \item Text \anskey{answer}
  \item Text
  \begin{keyanspic}
    ...
  \end{keyanspic}
\end{enumext}
```

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

The optional arguments of the inner levels or the nested environment are filtered by excluding all  $\langle keys \rangle$  related to the “*stored system*” along with the keys `series`, `resume` and `resume*` when storing in  $\langle sequence \rangle$ .

### 6.1 Keys for storage system

- The only  $\langle keys \rangle$  available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the  $\langle keys \rangle$  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  $\langle sequence \rangle$  and  $\langle prop list \rangle$  in which the contents will be “*stored*” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the  $\langle sequence \rangle$  or  $\langle prop list \rangle$  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 arguments of the inner levels or nested environment that will be passed to the  $\langle sequence \rangle$ . The  $\langle key list \rangle$  passed to this key ignores any  $\langle keys \rangle$  in the “*stored system*” 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  $\langle keys \rangle$  that will be stored by default in the  $\langle sequence \rangle$  would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the  $\langle sequence \rangle$  only the key `columns=3` ignoring all the others.

`save-sep = {\text symbol}` default:  $\{, \}$

Sets the *text symbol* that will separate the current  $\langle label \rangle$  to the *optional argument* passed to the `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments and storing them in the  $\langle store name \rangle$  defined by the `save-ans` key. The  $\{\text{text symbol}\}$  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  $\langle store name \rangle$  set by `save-ans` key. To reference the location of the “*stored content*” within the environment you must use `\ref{\store name : position}`, where  $\langle position \rangle$  corresponds to the position occupied by the “*stored content*” in the  $\langle store name \rangle$  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` within the environment in which the key `save-ans=test` was set.

`mark-ref = {\symbol}` default: `\textasteriskcentered`

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 L<sup>A</sup>T<sub>E</sub>X 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*` = { $\langle symbol \rangle$ } default:  $\$star$   
 Sets the *symbol* for `\item*` when using the key `item-star` and stores { $\langle content \rangle$ } in the *sequence* { $\langle store name \rangle$ } of the form `\item*[\langle symbol \rangle] \langle content \rangle`. The *symbol* can be in text or math mode, for example `item-sym*={\ast}` stores `\item*[\ast] \langle content \rangle`.

`item-pos*` = { $\langle rigid length \rangle$ } default: *not set*  
 Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores { $\langle content \rangle$ } in the *sequence* { $\langle store name \rangle$ } of the form `\item*[\langle symbol \rangle][\langle offset \rangle] \langle content \rangle`.

### 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.<br>* <input type="text" value="first answer"/><br>2. Text containing our instructions or questions.<br>(a) Question.<br>* <input type="text" value="second answer"/> | 3. Text containing our instructions or questions.<br>* <input type="text" value="third answer"/><br>4. Text containing our instructions or questions.<br>* <input type="text" value="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* { $\langle store name \rangle$ } 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  $\text{\LaTeX}$  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]
      \first answer
    \end{anskey*}
\end{enumext}
```

```

\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.

[5] First answer with verbatim

6. Text containing our instructions or questions.

(a) Question.

[6] second answer

7. Text containing our instructions or questions.

[7] third answer

8. Text containing our instructions or questions.

[8] fourth answer

## 6.4 The environments `keyans` and `keyans*`

```

keyans \begin{keyans}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}
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 `\langle content \rangle` (if it is present) 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
  \end{keyans*}
\end{enumext}

```



```
\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*[\textit{note}] Correct choice
\miniright
\includegraphics[scale=0.25]{example-image-a}
Some text
\end{keyans}
\end{enumext}
```

1. Text containing a question.  
A) Choice  
C) Choice  
E) Choice
- \* B) Correct choice  
D) Choice
2. Text containing a question and image.  
A) Choice  
B) Choice  
C) Choice  
D) Choice  
\* E) [note] Correct choice
-   
Some text

6.5 The environment keyanspic

```
keyanspic \begin{keyanspic}[\langle n^{\circ} above, n^{\circ} below \rangle]{\anspic{\langle drawing \rangle}\anspic*[\langle content \rangle]{\langle drawing \rangle}}
```

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

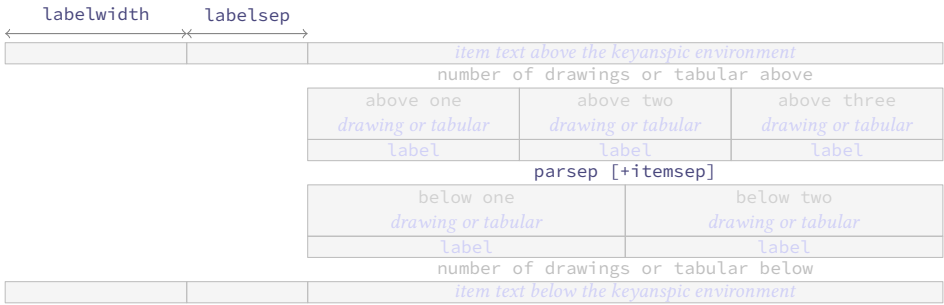


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

The optional argument determines the number drawings or tabular “above” and “below” within the environment. The vertical separation between “above” and “below” is controlled by the values set by `parsep` and `itemsep` keys passed to `keyans` environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

6.5.1 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 `\content` (if it is present) in *sequence* and *prop list* `{\store name}` 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.

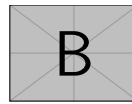
Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
\item Question with images.
\begin{keyanspic}[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*[\textit{note}]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}
\end{enumext}
```

1. Question with images.



A)



B)



C)



D)



\* E)[note]

## 6.6 Printing stored content

### 6.6.1 The command `\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 [<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

`sep = {<code>}` default: *empty*  
 Establishes the separation between *each* content stored in *prop list* `{<store name>}`. For example, you can use `sep={\ [10pt]}` for vertical separation of stored contents.

`step = {<integer>}` default: **1**  
 Sets the increment (`<step>`) applied to the value set by key `start` for each element stored in *prop list* `{<store name>}`. The value must be a *positive integer*.

`start = {<integer>}` default: **1**  
 Sets the *position* of the *prop list* `{<store name>}` from which execution will start. The value must be a *positive integer*.

`stop = {<integer>}` default: **0**  
 Sets the *position* of the *prop list* `{<store name>}` from which execution it will finish executing. The value must be a *positive integer*.

`before = {<code>}` default: *empty*  
 Sets the `{<code>}` that will be executed *before* each content stored in *prop list* `{<store name>}`. The `{<code>}` must be passed between braces.

`after = {<code>}` default: *empty*  
 Sets the `{<code>}` that will be executed *after* each content stored in *prop list* `{<store name>}`. The `{<code>}` must be passed between braces.

`wrapper = {<code> {#1} more code}` default: *empty*  
 Wraps the content stored in *prop list* `{<store name>}` referenced by `{#1}`. The `{<code>}` must be passed between braces. For example `\foreachkeyans[wrapper={\makebox[1em][l]{#1}}]{<store name>}`.

### 6.6.3 The command `\printkeyans`

---

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

---

The command `\printkeyans` prints “all stored content” in *sequence* `{<store name>}` defined by `save-ans` key placing this inside the `enumext` environment 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*  $\{\langle store\ name\rangle\}$  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 `\printkeyans*{\langle store\ name\rangle}` and the *sequence*  $\{\langle store\ name\rangle\}$  already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeyans*{\langle store\ name\rangle}` and the *sequence*  $\{\langle store\ name\rangle\}$  contains any `enumext` environments, they will start with the  $\langle keys\rangle$  set for the first level unless they are set in the optional argument or `save-key` is used to modify it.
- If we execute `\printkeyans{\langle store\ name\rangle}` and the *sequence*  $\{\langle store\ name\rangle\}$  contains any environment `enumext*`, they will start with the  $\langle keys\rangle$  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 `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[\langle print\rangle,\langle i\rangle]{\langle keys\rangle}` and `\setenumext[\langle print*\rangle]{\langle keys\rangle}`. If we need to set the  $\langle keys\rangle$  for the environment `enumext` “saved” in the *sequence*  $\{\langle store\ name\rangle\}$  we will use `\setenumext[\langle print\rangle,\langle level\rangle]{\langle keys\rangle}` and if we need to set the  $\langle keys\rangle$  for the environment `enumext*` “saved” in the *sequence*  $\{\langle store\ name\rangle\}$  we will use `\setenumext[\langle print\rangle,\langle *\rangle]{\langle keys\rangle}`.

Example

```
\begin{enumext}[save-ans=sample,columns=2,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) You use linux?

[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:
 

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:  $1 \text{ \AA} = 1 \times 10^{-10}$  m) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15}$  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. La velocità di  $1,00 \times 10^2$  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:  $1 \text{ \AA} = 1 \times 10^{-10}$  m) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15}$  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

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

### 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  $1,00 \times 10^2$  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:  $1 \text{ \AA} = 1 \times 10^{-10}$  m) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15}$  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. La velocità di  $1,00 \times 10^2$  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:  $1 \text{ \AA} = 1 \times 10^{-10}$  m) e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15}$  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 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.  $2\alpha + 2\delta = 90^\circ$

---

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

- 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)  $2\alpha + 2\delta = 90^\circ$

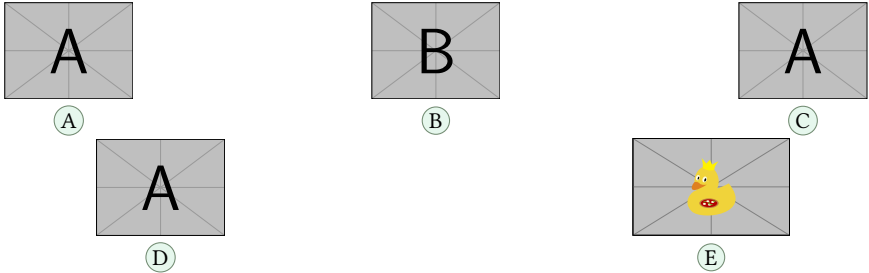
(2)  $\angle EDF = 45^\circ$

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) ~~ETX~~e 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.  $(x - 1)^2$

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

\*

Example 5

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

1

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request

from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character for pursuing an unexpected course of action.

2





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={-\labelwidth - 4pt}`.

- 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.

If we add `align=right` it will look like this:

- 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.

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*` key 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 *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.  
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.
- 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.

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.

## 9 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  $\text{\LaTeX}$  contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The  $\text{\LaTeX}$  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. “ $\text{\LaTeX} 2_{\epsilon}$ : An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The  $\text{\LaTeX}$  Project. “Extensive support for hypertext in  $\text{\LaTeX}$ ”. 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  $\text{\LaTeX}$  Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The  $\text{\LaTeX}$  Project. “The  $\text{\LaTeX} 3$  Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The  $\text{\LaTeX}$  Project. “The  $\text{\LaTeX} 2_{\epsilon}$  sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.
- [13] The  $\text{\LaTeX}$  Project. “ $\text{\LaTeX}$  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.

## 10 Change history

**v1.0** 2024-06-29 – First public release.

11 Index of Documentation

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

C

Document class:

article ..... 2

book ..... 2

exam ..... 2

letter ..... 2

report ..... 2

\columnbreak ..... 4, 12

\columnsep ..... 10

Commands provide by enumext:

\anskey ..... 11–13

\anspic ..... 11, 12, 15

\foreachkeyans ..... 16

\getkeyans ..... 12, 16

\item\* ..... 5–7, 11, 12, 14, 15

\item ..... 5–10, 12, 14

\miniright ..... 10

\printkeyans ..... 6, 11, 16

\setenumextmeta ..... 6

\setenumext ..... 5–7, 11, 12, 14, 17

Counters defined by enumext:

enumXiii ..... 4

enumXii ..... 4

enumXiv ..... 4

enumXi ..... 4

enumXviii ..... 4

enumXvii ..... 4

enumXvi ..... 4

enumXv ..... 4

E

Environments provide by enumext:

anskey\* ..... 11–13

enumext\* ..... 4–14, 16, 17

enumext ..... 4–14, 16, 17, 20

keyans\* ..... 4–14

keyanspic ..... 4, 7, 8, 11–13, 15, 20

keyans ..... 4–9, 11–15, 20

Environments:

Verbatim ..... 13

enumerate ..... 1, 3, 5, 21

figure ..... 5

list ..... 3, 9, 22

minipage ..... 3–5, 10, 22

multicols ..... 3, 4, 10

table ..... 5

task ..... 5

F

\footnote ..... 5

I

\itemsep ..... 8

K

Keys for \anskey provide by enumext:

break-col ..... 12

item-join ..... 12

item-pos\* ..... 13

item-star ..... 12, 13

item-sym\* ..... 13

Keys for \foreachkeyans provide by enumext:

after ..... 16

before ..... 16

sep ..... 16

start ..... 16

step ..... 16

stop ..... 16

wrapper ..... 16

Keys for anskey\* provide by enumext:

break-col ..... 12

force-eol ..... 13

item-join ..... 12

item-pos\* ..... 13

item-star ..... 12, 13

item-sym\* ..... 13

overwrite ..... 13

write-env ..... 13

Keys for environments provide by enumext:

above\* ..... 8

above ..... 8

after ..... 9, 10

align ..... 7, 21

base-fix ..... 8

before\* ..... 9

before ..... 9

below\* ..... 9

below ..... 8

check-ans ..... 12

columns-sep ..... 4, 10

columns ..... 4, 8, 10

first ..... 9

font ..... 7

item-pos\* ..... 5, 6

item-sym\* ..... 5, 6

itemindent ..... 9

itemsep ..... 8, 15

labelsep ..... 3–7, 9, 10, 12, 20, 21

labelwidth ..... 3, 4, 6, 7, 9, 10, 12, 20, 21

labelwith ..... 5

label ..... 7, 9, 14, 20, 21

list-indent ..... 3, 9

list-offset ..... 3, 9, 21

listparindent ..... 9

mark-ans ..... 12

mark-pos ..... 12

mark-ref ..... 11

mini-env ..... 4, 5, 8, 10

mini-right\* ..... 7, 10

mini-right ..... 7, 10

mini-sep ..... 4, 10

no-store ..... 11–13

noitemsep ..... 8

nosep ..... 8, 20

overwrite ..... 13

parsep ..... 8, 15

partopsep ..... 8

ref ..... 4, 7

resume\* ..... 7, 10, 11

resume ..... 7, 10, 11

rightmargin ..... 9

save-ans ..... 4, 6, 10–16

save-key .....	10, 11, 17	\linewidth .....	10
save-ref .....	4, 7, 11-13, 16	\listparindent .....	9
save-sep .....	11		
series .....	7, 10, 11	<b>P</b>	
show-ans .....	11, 12	Packages:	
show-length .....	8	enumerate .....	21
show-pos .....	11, 12, 16	enumext .....	1-5, 7, 15, 21
start* .....	9, 10	enumitem .....	3-5, 9, 21
start .....	9, 10	fancyvrb .....	13
topsep .....	8, 9	footnotehyper .....	5
widest .....	7	hyperref .....	4, 5, 11-13, 21, 22
wrap-ans .....	12	l3keys .....	7
wrap-label* .....	8, 21	l3prop .....	1, 21
wrap-label .....	7, 8	l3seq .....	1, 21
wrap-opt .....	12	multicol .....	1, 2, 4, 22
write-env .....	13	scontents .....	1, 2, 13
		task .....	5, 6
<b>L</b>		xsim .....	2
\label .....	4	\parsep .....	8
Labels provide by enumext:		\partopsep .....	8
\Alph* .....	7, 14		
\Roman* .....	7	<b>R</b>	
\alph* .....	7	\raggedcolumns .....	4
\arabic* .....	7	\ref .....	4
\roman* .....	7	\rightmargin .....	9
\labelsep .....	3, 7		
\labelwidth .....	3, 7	<b>T</b>	
		\topsep .....	8

## 12 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.

### 12.1 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.

### 12.2 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>
```

### 12.3 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   {2024-06-29}
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 }
```

### 12.4 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:` (§12.5.1).

```

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` (§12.10) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§12.13).

```

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 (§12.13).

```

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 (§12.24).

```

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` (§12.14) and `label` (§12.12) 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 (§12.17). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace`:NNNNNNNNNN (§12.37.1).

```
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
```

Internal variables used by `columns` key §12.21).

```
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 }
85 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_multicols_above_X_skip`, `\l__enumext_multicols_below_X_skip`, and `\g__enumext_multicols_right_X_skip`.)

```
\g__enumext_minipage_stat_int
\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 (§12.22.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§12.20, §12.22).

```
86 \int_new:N \g__enumext_minipage_stat_int
87 \skip_new:N \l__enumext_minipage_left_skip
88 \skip_new:N \l__enumext_minipage_right_skip
89 \skip_new:N \l__enumext_minipage_after_skip
90 \skip_new:N \g__enumext_minipage_right_skip
91 \skip_new:N \g__enumext_minipage_after_skip
92 \cs_set_protected:Npn \__enumext_tmp:n #1
93 {
94   \dim_new:c { \l__enumext_minipage_left_#1_dim }
95   \bool_new:c { \l__enumext_minipage_active_#1_bool }
96 }
97 \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 (§12.12), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§12.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§12.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§12.12). 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 (§12.19).

```
98 \cs_set_protected:Npn \__enumext_tmp:n #1
99 {
100   \bool_new:c { \l__enumext_wrap_label_#1_bool }
101   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
102   \int_new:c { \l__enumext_start_#1_int }
103   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
104   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
105   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
106   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
107   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
108 }
109 \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 (§12.25.1) 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` (§12.29) 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*` (§12.30).

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*` (§12.36) and `\anspic*` (§12.40.1) 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.

```

110 \bool_new:N \l__enumext_store_active_bool
111 \tl_new:N \l__enumext_store_name_tl
112 \tl_new:N \g__enumext_store_name_tl
113 \tl_new:N \l__enumext_store_anskey_arg_tl
114 \tl_new:N \l__enumext_store_anskey_env_tl
115 \tl_new:N \l__enumext_store_anskey_opt_tl
116 \tl_new:N \l__enumext_store_current_label_tl
117 \tl_new:N \l__enumext_store_current_opt_arg_tl
118 \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` (§12.47).

```

119 \tl_new:N \l__enumext_setkey_tmpa_tl
120 \tl_new:N \l__enumext_setkey_tmpb_tl
121 \int_new:N \l__enumext_setkey_tmpa_int
122 \seq_new:N \l__enumext_setkey_tmpa_seq
123 \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
\l__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§12.46) and `\foreachkeyans` command (§12.49).

```

124 \tl_new:N \l__enumext_meta_path_tl
125 \seq_new:N \l__enumext_foreach_print_seq
126 \tl_new:N \l__enumext_foreach_name_prop_tl
127 \tl_new:N \l__enumext_foreach_default_keys_tl

```

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

```

\l__enumext_print_keyans_starred_tl
\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` (§12.46), `show-pos` key (§12.26), `item-sym*` key (§12.34), `save-key` key (§12.26.2) and “*storage level system*”.

```

128 \tl_new:N \l__enumext_print_keyans_starred_tl
129 \str_new:N \l__enumext_mark_position_str
130 \tl_new:N \g__enumext_item_symbol_aux_tl
131 \cs_set_protected:Npn \__enumext_tmp:n #1
132 {
133   \tl_new:c { \l__enumext_print_keyans_#1_tl }
134   \tl_new:c { \l__enumext_store_save_key_#1_tl }
135   \bool_new:c { \l__enumext_store_save_key_#1_bool }
136   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
137 }
138 \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_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

```

Internal variables used by `keyanspic` environment (§12.40.2).

```

139 \seq_new:N \l__enumext_keyans_pic_body_seq
140 \dim_new:N \l__enumext_keyans_pic_width_dim
141 \int_new:N \l__enumext_keyans_pic_above_int
142 \int_new:N \l__enumext_keyans_pic_below_int
143 \skip_new:N \l__enumext_keyans_pic_above_skip

```

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

```

\l__enumext_check_answers_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int
\g__enumext_item_number_bool
\g__enumext_item_answer_diff_int

```

Internal variables used by “*internal check answer*” mechanism (§12.25.3) 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.

```

144 \bool_new:N \l__enumext_check_answers_bool
145 \bool_new:N \g__enumext_check_ans_key_bool
146 \tl_new:N \l__enumext_check_start_line_env_tl
147 \int_new:N \g__enumext_check_starred_cmd_int
148 \int_new:N \g__enumext_item_anskey_int
149 \int_new:N \g__enumext_item_number_int
150 \bool_new:N \l__enumext_item_number_bool
151 \int_new:N \g__enumext_item_answer_diff_int

```

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

```

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

```

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§12.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

152 \bool_new:N \l__enumext_hyperref_bool
153 \bool_new:N \l__enumext_footnotes_key_bool

```

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

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

Internal variables used by `save-ref` key (§12.26). 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` (§12.8) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

154 \tl_new:N \l__enumext_newlabel_arg_one_tl
155 \tl_new:N \l__enumext_newlabel_arg_two_tl
156 \tl_new:N \l__enumext_write_aux_file_tl
157 \cs_set_protected:Npn \__enumext_tmp:n #1
158 {
159   \tl_new:c { \l__enumext_label_copy_#1_tl }
160 }
161 \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.)

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

Internal variables used for redefinition of `\footnote` (§12.42).

```

162 \int_new:N \g__enumext_footnote_int
163 \seq_new:N \g__enumext_footnote_arg_seq
164 \seq_new:N \g__enumext_footnote_int_seq

```

(End of definition for `\g__enumext_footnote_int`, `\g__enumext_footnote_arg_seq`, and `\g__enumext_footnote_int_seq`.)

```

\l__enumext_item_starred_X_bool
\l__enumext_item_column_pos_X_int
\g__enumext_item_count_all_X_int
\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_tmpa_X_dim
\l__enumext_item_text_X_box
\l__enumext_joined_width_X_dim
\l__enumext_item_width_X_dim
\g__enumext_item_symbol_aux_X_tl
\l__enumext_align_label_X_str
\g__enumext_minipage_active_X_bool
\l__enumext_miniright_code_X_box
\g__enumext_minipage_center_X_bool
\g__enumext_minipage_right_X_dim
\g__enumext_minipage_right_X_skip

```

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

```

165 \cs_set_protected:Npn \__enumext_tmp:n #1
166 {
167   \bool_new:c { \l__enumext_item_starred_#1_bool }
168   \int_new:c { \l__enumext_item_column_pos_#1_int }
169   \int_new:c { \g__enumext_item_count_all_#1_int }
170   \int_new:c { \l__enumext_joined_item_#1_int }
171   \int_new:c { \l__enumext_joined_item_aux_#1_int }
172   \int_new:c { \l__enumext_tmpa_#1_int }
173   \dim_new:c { \l__enumext_tmpa_#1_dim }
174   \box_new:c { \l__enumext_item_text_#1_box }
175   \dim_new:c { \l__enumext_joined_width_#1_dim }
176   \dim_new:c { \l__enumext_item_width_#1_dim }
177   \tl_new:c { \g__enumext_item_symbol_aux_#1_tl }
178   \str_new:c { \l__enumext_align_label_#1_str }
179   \bool_new:c { \g__enumext_minipage_active_#1_bool }
180   \box_new:c { \l__enumext_miniright_code_#1_box }
181   \bool_new:c { \g__enumext_minipage_center_#1_bool }
182   \dim_new:c { \g__enumext_minipage_right_#1_dim }
183   \skip_new:c { \g__enumext_minipage_right_#1_skip }
184 }
185 \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`.

```

186 \clist_const:Nn \c__enumext_all_envs_clist
187 {
188   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
189   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
190 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

## 12.5 Some utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§12.46) and `\foreachkeyans` command (§12.49).

```

191 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
192 \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.

```

193 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
194 {
195   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
196 }

```

(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`

```

197 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
198 {
199   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
200 }
201 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
202 {
203   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
204 }

```

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

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

```

205 \cs_new:Nn \__enumext_level:
206 {
207   \int_to_roman:n { \__enumext_level_int }
208 }

```

(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. 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?](#)

`\__enumext_if_is_int:nF`

`\__enumext_if_is_int:nTF`

```

209 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
210 {
211   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
212   { \prg_return_true: }
213   { \prg_return_false: }
214 }

```

(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.

```

215 \cs_new_protected:Nn \__enumext_regex_counter_style:
216 {
217   \tl_map_inline:Nn \c__enumext_counter_style_tl
218   {
219     \regex_replace_once:nnN { \c{##1}\* }
220     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
221   }
222 }

```

(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.

```

223 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
224 {
225     * ~ #2
226     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
227     = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
228 }

```

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

### 12.5.1 Utilities for environments and levels

`\__enumext_is_not_nested:`  
`\__enumext_is_on_first_level:`

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 nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

```

229 \cs_new_protected:Nn \__enumext_is_not_nested:
230 {
231     \str_case:en { \@currentenvir }
232     {
233         {enumext}
234         {
235             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
236             \bool_lazy_and:nnT
237             { \bool_not_p:n { \g__enumext_standar_bool } }
238             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
239             {
240                 \bool_gset_true:N \g__enumext_standar_bool
241             }
242         }
243         {enumext*}
244         {
245             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
246             \bool_lazy_and:nnT
247             { \bool_not_p:n { \g__enumext_starred_bool } }
248             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
249             {
250                 \bool_gset_true:N \g__enumext_starred_bool
251             }
252         }
253     }
254 }

```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§12.25.1), `\l__enumext_starred_first_bool` (§12.25.1) and `\l__enumext_anskey_env_bool` (§12.30) 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.

```

255 \cs_new_protected:Nn \__enumext_is_on_first_level:
256 {
257     \bool_lazy_all:nT
258     {
259         { \bool_if_p:N \g__enumext_standar_bool }
260         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
261         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
262     }
263     {
264         \bool_set_true:N \l__enumext_standar_first_bool
265         \bool_set_true:N \l__enumext_anskey_env_bool
266         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
267         \tl_gset:Nn \g__enumext_start_line_tl
268         {
269             on ~ line ~ \exp_not:V \inputlineno
270         }
271     }
272     \bool_lazy_all:nT
273     {
274         { \bool_if_p:N \g__enumext_starred_bool }
275         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }

```

```

276     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
277   }
278   {
279     \bool_set_true:N \l__enumext_starred_first_bool
280     \bool_set_true:N \l__enumext_anskey_env_bool
281     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
282     \tl_gset:Ne \g__enumext_start_line_tl
283       {
284         on ~ line ~ \exp_not:V \inputlineno
285       }
286   }
287 }

```

(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.

```

288 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
289 {
290   \str_case:en { \@currenvir }
291   {
292     {keyans}
293     {
294       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
295       \tl_set:Ne \l__enumext_check_start_line_env_tl
296         {
297           in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
298         }
299     }
300     {keyans*}
301     {
302       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
303       \tl_set:Ne \l__enumext_check_start_line_env_tl
304         {
305           in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
306         }
307     }
308     {keyanspic}
309     {
310       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
311       \tl_set:Ne \l__enumext_check_start_line_env_tl
312         {
313           in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
314         }
315     }
316   }
317 }

```

(End of definition for `\__enumext_keyans_name_and_start:`)

### 12.5.2 Utilities for log and terminal

`\__enumext_reset_global_vars:`

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.

`\__enumext_reset_global_int:`

`\__enumext_reset_global_bool:`

`\__enumext_reset_global_tl:`

```

318 \cs_new_protected:Nn \__enumext_reset_global_vars:
319 {
320   \__enumext_reset_global_int:
321   \__enumext_reset_global_bool:
322   \__enumext_reset_global_tl:
323 }
324 \cs_new_protected:Nn \__enumext_reset_global_int:
325 {
326   \int_gzero:N \g__enumext_item_number_int
327   \int_gzero:N \g__enumext_item_anskey_int
328   \int_gzero:N \g__enumext_item_answer_diff_int
329 }
330 \cs_new_protected:Nn \__enumext_reset_global_bool:
331 {
332   \bool_gset_false:N \g__enumext_check_ans_key_bool
333   \bool_gset_false:N \g__enumext_standar_bool
334   \bool_gset_false:N \g__enumext_starred_bool

```



```

335   }
336   \cs_new_protected:Nn \__enumext_reset_global_tl:
337   {
338     \tl_gclear:N \g__enumext_store_name_tl
339     \tl_gclear:N \g__enumext_start_line_tl
340     \tl_gclear:N \g__enumext_envir_name_tl
341   }

```

(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.

```

342   \cs_new_protected:Nn \__enumext_log_global_vars:
343   {
344     \msg_log:nneeee { enumext } { prop-seq-int-hook }
345     { \g__enumext_store_name_tl }
346     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
347     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
348     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
349   }

```

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.

```

350   \cs_new_protected:Nn \__enumext_log_answer_vars:
351   {
352     \msg_log:nneeee { enumext } { item-answer-hook }
353     { \int_use:N \g__enumext_item_number_int }
354     { \int_use:N \g__enumext_item_anskey_int }
355     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
356   }

```

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

## 12.6 Copying list and minipage environments

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

```

\list{<arg one>}{<arg two>}
  \item[<opt>]
\endlist

```

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

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

```

357   \__enumext_at_begin_document:n
358   {
359     \cs_new_eq:NN \__enumext_start_list:nn \list
360     \cs_new_eq:NN \__enumext_stop_list: \endlist
361     \cs_new_eq:NN \__enumext_item_std:w \item
362   }

```

(End of definition for `\__enumext_start_list:nn`, `\__enumext_stop_list:`, and `\__enumext_item_std:w`.)

The `minipage` environment provided by  $\LaTeX$  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 `minipage` environment or a related command.

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

```

363   \__enumext_at_begin_document:n
364   {
365     \cs_new_eq:NN \__enumext_minipage:w \minipage
366     \cs_new_eq:NN \__enumext_endminipage: \endminipage
367   }

```

(End of definition for `\__enumext_minipage:w` and `\__enumext_endminipage:`.)

## 12.7 The internal minipage environment

```
\__enumext_internal_mini_page:
  __enumext_mini_env*
```

The function `\__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` 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. This function is passed to the function `\__enumext_safe_exec:` in the `enumext` environment definition (§12.38) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§12.43)

```
368 \cs_new_protected:Nn \__enumext_internal_mini_page:
369 {
370   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
371   {
372     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
373     {
374       \__enumext_minipage:w [ t ] { ##1 }
375       \legacy_if_gset_false:n { @minipage }
376       \skip_vertical:N \c_zero_skip
377     }
378     {
379       \skip_vertical:N \c_zero_skip
380       \__enumext_endminipage:
381     }
382   }
383 }
```

(End of definition for `\__enumext_internal_mini_page:` and `__enumext_mini_env*`.)

## 12.8 Compatibility with hyperref and footnotehyper

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

```
384 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
385 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

```
\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:
```

The function `\__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyprerref_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 `\__enumext_footnotes_key_bool` to “*true*”.

```
386 \cs_new_protected:Nn \__enumext_after_hyperref:
387 {
388   \IfPackageLoadedTF { hyperref }
389   {
390     \msg_info:nnn { enumext } { package-load } { hyperref }
391     \bool_set_true:N \l__enumext_hyprerref_bool
392     \IfHyperBoolean{hyperfootnotes}
393     {
394       \typeout{hyperfootnotes=true}
395       \bool_set_true:N \l__enumext_footnotes_key_bool
396     }
397     { \typeout{hyperfootnotes=false} }
398   }
399   { }
```

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`.

```
400   \bool_if:NT \l__enumext_footnotes_key_bool
401   {
402     \IfPackageLoadedTF { footnotehyper }
403     {
404       \msg_info:nnn { enumext } { package-load } { footnotehyper }
405     }
406     {
407       \typeout{No ~ footnotehyper ~ load}
408       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
409       \bool_set_false:N \l__enumext_footnotes_key_bool
410     }
411   }
```

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

```

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

(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: `\__enumext_newlabel_arg_one_tl`  
 #2: `\__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.

```

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

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

## 12.9 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.

```

437 \dim_zero_new:N \itemwidth
```

## 12.10 Definition of counters

`\__enumext_define_counters:Nn`  
`\__enumext_define_counters:cn`

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 `\__enumext_counter_X_tl` for “store” the counter’s name.  
 #2: The counter’s name.

```

438 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
439 {
440   \cs_if_exist:cTF { c@ #2 }
441   { \msg_fatal:nnn { enumext } { counters } { #2 } }
442   {
443     \tl_set:Nn #1 { #2 }
444     \newcounter { #2 }
445   }
446 }
```

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

```

enumXi    The counters created here are enumXi, enumXii, enumXiii and enumXiv for enumext environment, enumXv
enumXii   for keyans environment, enumXvi for keyanspic environment, enumXvii for enumext* and enumXviii
enumXiii  for the keyans* environments.
enumXiv   447 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv    448 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi   449 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii  450 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 451 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
          452 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
          453 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
          454 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

(End of definition for enumXi and others.)

## 12.11 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` environment and the `keyans` environment, so it is necessary to get a default value for `labelwidth` from these *labels* at the same time.

```

455 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
456 {
457   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
458   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
459 }
460 \__enumext_register_counter_style:Nn \arabic { 0 }
461 \__enumext_register_counter_style:Nn \Alph { M }
462 \__enumext_register_counter_style:Nn \alph { m }
463 \__enumext_register_counter_style:Nn \Roman { VIII }
464 \__enumext_register_counter_style:Nn \roman { viii }

```

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

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

`\__enumext_label_width_by_box:cv`

```

465 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
466 {
467   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
468   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
469 }
470 \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` 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.

`\__enumext_label_style:cvn`

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.

```

471 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
472 {
473   \tl_clear_new:N #1
474   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
475   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
476   \tl_map_inline:Nn \g__enumext_counter_styles_tl
477   {
478     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
479     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
480     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
481   }
482   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
483   { \tl_use:N \g__enumext_widest_label_tl }
484   \tl_set_eq:cN { the #2 } #1
485 }
486 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

(End of definition for `\__enumext_label_style:Nnn`.)

## 12.12 Setting keys associated with label

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

```

labelsep
labelwidth
wrap-label
wrap-label*
487 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
488 {
489   \keys_define:nn { enumext / #1 }
490   {
491     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
492     font .value_required:n = true,
493     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
494     labelsep .initial:n = {0.3333em},
495     labelsep .value_required:n = true,
496     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
497     labelwidth .value_required:n = true,
498     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
499     wrap-label .initial:n = {##1},
500     wrap-label .value_required:n = true,
501     wrap-label* .code:n = {
502       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
503       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
504     },
505     wrap-label* .value_required:n = true,
506   }
507 }
508 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

- In this point, the following are set `\__enumext_wrapper_label_X:n` which will be used by `\__enumext_make_label:` for the different levels of the `enumext` environment and is set to `\__enumext_wrapper_label_v:n` which will be used by `\__enumext_keyans_make_label:` for `keyans` and `keyanspic` environments.

`align` The `align` key is implemented differently for “starred” and “non starred” environments.

```

509 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
510 {
511   \keys_define:nn { enumext / #1 }
512   {
513     align .choice:,
514     align / left .code:n =
515       {
516         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
517         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
518       },
519     align / right .code:n =
520       {
521         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
522         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
523       },
524     align / center .code:n =
525       {
526         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
527         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
528       },
529     align / unknown .code:n =
530       \msg_error:nneee { enumext } { unknown-choice }
531       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
532     align .initial:n = left,
533     align .value_required:n = true,
534   }
535 }
536 \clist_map_inline:nn
537 {
538   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
539 }
540 { \__enumext_tmp:nn #1 }

541 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
542 {
543   \keys_define:nn { enumext / #1 }
544   {
545     align .choice:,
```

```

546 align / left .code:n = \str_set:cn { l__enumext_align_label#2_str } { l },
547 align / right .code:n = \str_set:cn { l__enumext_align_label#2_str } { r },
548 align / center .code:n = \str_set:cn { l__enumext_align_label#2_str } { c },
549 align / unknown .code:n =
550     \msg_error:nneee { enumext } { unknown-choice }
551     { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
552 align .initial:n = left,
553 align .value_required:n = true,
554 }
555 }
556 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

## 12.13 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 `<label>`, the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “label and ref” system.

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

Here we set the default `<labels>` of the *four levels* of `enumext` environment, along with the default value for `labelwidth` key and `ref` key.

```

label \l__enumext_label_i_tl 557 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
ref   \l__enumext_label_ii_tl 558 {
\l__enumext_label_iii_tl 559     \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 560     {
561         label .code:n = {
562             \__enumext_label_style:cnv { l__enumext_label#2_tl }
563             { l__enumext_counter#2_tl } {##1}
564             \dim_set_eq:cN { l__enumext_labelwidth#2_dim }
565             \l__enumext_current_widest_dim
566         },
567         label .initial:n = #3,
568         label .value_required:n = true,
569         ref .code:n = \__enumext_standar_ref:n {##1},
570         ref .value_required:n = true,
571     }
572 }
573 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
574 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
575 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
576 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

(End of definition for label and others.)

`\__enumext_standar_ref:n` 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.

```

577 \cs_new_protected:Npn \__enumext_standar_ref:n #1
578 {
579     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
580     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
581     {
582         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
583     }
584     {
585         \tl_set_eq:Nc
586             \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
587         \__enumext_regex_counter_style:
588         \tl_set_eq:Nc
589             \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
590         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
591         {
592             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
593             { \exp_not:V \l__enumext_ref_key_arg_tl }
594         }
595     }
596 }

```



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

```

597 \cs_new_protected:Nn \__enumext_standar_ref:
598 {
599   \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
600   {
601     \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
602   }
603 }

```

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

### 12.13.2 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.

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
604 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
605 {
606   \keys_define:nn { enumext / #1 }
607   {
608     label .code:n = {
609       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
610       { \__enumext_counter_#2_tl } {##1}
611       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
612       \__enumext_current_widest_dim
613     },
614     label .initial:n = #3,
615     label .value_required:n = true,
616     ref .code:n = \__enumext_starred_ref:n {##1},
617     ref .value_required:n = true,
618   }
619 }
620 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
621 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

The implementation of `\__enumext_starred_ref:n` is the same as that used for the environment `enumext`.

```

622 \cs_new_protected:Npn \__enumext_starred_ref:n #1
623 {
624   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
625   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
626   {
627     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
628     {
629       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
630     }
631     {
632       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
633       \__enumext_regex_counter_style:
634       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
635       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
636       {
637         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
638         { \exp_not:V \l__enumext_ref_key_arg_tl }
639       }
640     }
641   }
642   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
643   {
644     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
645     {
646       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
647     }
648     {
649       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
650       \__enumext_regex_counter_style:
651       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
652       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
653       {
654         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
655         { \exp_not:V \l__enumext_ref_key_arg_tl }

```

```

656         }
657     }
658 }
659 }

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.

660 \cs_new_protected:Nn \__enumext_starred_ref:
661 {
662     \int_compare:nNt { \__enumext_level_h_int } = { 1 }
663     {
664         \tl_if_empty:NF \__enumext_renew_the_count_vii_tl
665         {
666             \tl_use:N \__enumext_renew_the_count_vii_tl
667         }
668     }
669     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
670     {
671         \tl_if_empty:NF \__enumext_renew_the_count_viii_tl
672         {
673             \tl_use:N \__enumext_renew_the_count_viii_tl
674         }
675     }
676 }

```

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

### 12.13.3 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

677 \keys_define:nn { enumext / keyans }
678 {
679     label .code:n = {
680         \__enumext_label_style:cvn { \__enumext_label_v_tl }
681         { \__enumext_counter_v_tl } {#1}
682         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
683         \__enumext_current_widest_dim
684         \__enumext_label_style:cvn { \__enumext_label_vi_tl }
685         { \__enumext_counter_vi_tl } {#1}
686         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
687         \__enumext_current_widest_dim
688     },
689     label .initial:n = \Alph*,
690     label .value_required:n = true,
691     ref .code:n = \__enumext_keyans_ref:n {#1},
692     ref .value_required:n = true,
693 }

```

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

The implementation of `\__enumext_keyans_ref:n` is the same as that used for the environment `enumext`.

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:

694 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
695 {
696     \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
697     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
698     {
699         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
700     }
701     {
702         \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_v_tl
703         \__enumext_regex_counter_style:
704         \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
705         \tl_put_right:Ne \__enumext_renew_the_count_v_tl
706         {
707             \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
708             { \exp_not:V \__enumext_ref_key_arg_tl }
709         }
710     }
711 }

```

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

```

712 \cs_new_protected:Nn \__enumext_keyans_ref:
713 {
714   \tl_if_empty:NF \__enumext_renew_the_count_v_tl
715   {
716     \tl_use:N \__enumext_renew_the_count_v_tl
717   }
718 }

```

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

## 12.14 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: \__enumext_label_X_tl
#2: \__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.

```

719 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
720 {
721   \__enumext_if_is_int:nTF { #3 }
722   {
723     \int_set:Nn #2 {#3}
724   }
725   {
726     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
727     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
728     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
729     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
730   }
731 }
732 \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: \__enumext_label_X_tl
#3: \__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.

```

733 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
734 {
735   \__enumext_if_is_int:nTF {#4}
736   {
737     \setcounter{enumX#1} { #4 }
738   }
739   {
740     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
741     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
742     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
743     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
744   }
745   \__enumext_label_width_by_box:cv
746   { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
747 }
748 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for `\__enumext_widest_from:nNNn.`)

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

```

widest
start*
start
749 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
750 {

```

```

751 \keys_define:nn { enumext / #1 }
752 {
753   start* .code:n = {
754     \__enumext_start_from:ccn
755     { l__enumext_label_#2_tl }
756     { l__enumext_start_#2_int } {##1}
757   },
758   start* .value_required:n = true,
759   start .code:n = {
760     \__enumext_start_from:cce
761     { l__enumext_label_#2_tl }
762     { l__enumext_start_#2_int } { \int_eval:n {##1} }
763   },
764   start .initial:n = 1,
765   start .value_required:n = true,
766   widest .code:n = {
767     \__enumext_widest_from:nccn {#2}
768     { l__enumext_label_#2_tl }
769     { l__enumext_labelwidth_#2_dim } {##1}
770   },
771   widest .value_required:n = true,
772 }
773 }
774 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

## 12.15 Setting keys for vertical spaces

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

```

775 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
776 {
777   \keys_define:nn { enumext / #1 }
778   {
779     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
780     topsep .initial:n = {#3},
781     topsep .value_required:n = true,
782     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
783     partopsep .initial:n = {#4},
784     partopsep .value_required:n = true,
785     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
786     parsep .initial:n = {#5},
787     parsep .value_required:n = true,
788     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
789     itemsep .initial:n = {#6},
790     itemsep .value_required:n = true,
791     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
792     noitemsep .value_forbidden:n = true,
793     nosep .meta:n = {
794       itemsep = 0pt, parsep = 0pt,
795       topsep = 0pt, partopsep = 0pt,
796     },
797     nosep .value_forbidden:n = true,
798   }
799 }

```

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

```

800 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
801 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
802 { 4.0pt plus 2.0pt minus 1.0pt }
803 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
804 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
805 { 2.0pt plus 1.0pt minus 1.0pt }
806 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
807 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
808 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
809 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
810 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
811 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
812 { 2.0pt plus 1.0pt minus 1.0pt }
813 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }

```

```

814 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
815 { 4.0pt plus 2.0pt minus 1.0pt }
816 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
817 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
818 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

## 12.16 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:` and then apply `\vspace{-\baselineskip}` and set `topsep=0pt` for the “first level” of the nested `enumext` or `enumext*` environments.

```

base-fix
\__enumext_nested_base_line_fix:
819 \cs_set_protected:Npn \__enumext_tmp:n #1
820 {
821   \keys_define:nn { enumext / #1 }
822   {
823     base-fix .bool_set:N = \__enumext_base_line_fix_bool,
824     base-fix .initial:n = false,
825     base-fix .value_forbidden:n = true,
826   }
827 }
828 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

The function `\__enumext_nested_base_line_fix:` will be in charge of applying the baseline correction and adjusting the `\keys`. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§12.38) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§12.43)

```

829 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
830 {
831   \bool_lazy_and:nnT
832   { \bool_if_p:N \__enumext_standar_first_bool }
833   { \bool_if_p:N \__enumext_base_line_fix_bool }
834   {
835     \mode_leave_vertical:
836     \vspace { -\baselineskip }
837     \keys_set:nn { enumext / level-1 }
838     {
839       topsep = 0pt, above = 0pt, above* = 0pt,
840     }
841   }
842   \bool_lazy_and:nnT
843   { \bool_if_p:N \__enumext_starred_first_bool }
844   { \bool_if_p:N \__enumext_base_line_fix_bool }
845   {
846     \mode_leave_vertical:
847     \vspace { -\baselineskip }
848     \keys_set:nn { enumext / enumext* }
849     {
850       topsep = 0pt, above = 0pt, above* = 0pt,
851     }
852   }
853   \bool_set_false:N \__enumext_base_line_fix_bool
854 }

```

🚗 This key is enabled by default in the command `\printkeyans` (§12.46).

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

## 12.17 Setting keys for horizontal spaces

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

```

855 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
856 {
857   \keys_define:nn { enumext / #1 }
858   {
859     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
860     itemindent .value_required:n = true,
861     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },

```

```

862     rightmargin .value_required:n = true,
863     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
864     listparindent .value_required:n = true,
865     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
866     list-offset .value_required:n = true,
867     list-indent .code:n =
868         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
869         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
870     list-indent .value_required:n = true,
871 }
872 }
873 \clist_map_inline:nn
874 {
875     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
876 }
877 { 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.

```

878 \cs_set_protected:Npn l__enumext_tmp:nn #1 #2
879 {
880     \keys_define:nn { enumext / #1 }
881     {
882         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
883         itemindent .value_required:n = true,
884         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
885         rightmargin .value_required:n = true,
886         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
887         listparindent .value_required:n = true,
888         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
889         list-offset .value_required:n = true,
890         list-indent .meta:n = { list-offset = ##1 },
891         list-indent .value_required:n = true,
892     }
893 }
894 \clist_map_inline:nn
895 {
896     {enumext*}{vii}, {keyans*}{viii}
897 }
898 { l__enumext_tmp:nn #1 }

```

### 12.17.1 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.

```

899 \cs_set_protected:Nn l__enumext_fake_item:
900 {
901     \dim_compare:nNnT
902     { \dim_use:c { l__enumext_fake_item_indent_ l__enumext_level: _dim } }
903     >
904     { \c_zero_dim }
905     {
906         \tl_set:ce { l__enumext_fake_item_indent_ l__enumext_level: _tl }
907         {
908             \exp_not:N \mode_leave_vertical:
909             \exp_not:n { \skip_horizontal:n
910                 { \dim_use:c { l__enumext_fake_item_indent_ l__enumext_level: _dim } }
911             }
912             \ignorespaces
913         }
914     }
915 }
916 \cs_set_protected:Nn l__enumext_keyans_fake_item:
917 {
918     \dim_compare:nNnT
919     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
920     {
921         \tl_set:Nc l__enumext_fake_item_indent_v_tl
922         {

```



```

922         \exp_not:N \mode_leave_vertical:
923         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
924     }
925 }
926 }
927 \cs_set_protected:Nn \__enumext_fake_item_vii:
928 {
929     \dim_compare:nNnT
930     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
931     {
932         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
933         {
934             \exp_not:N \mode_leave_vertical:
935             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
936         }
937     }
938 }
939 \cs_set_protected:Nn \__enumext_fake_item_viii:
940 {
941     \dim_compare:nNnT
942     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
943     {
944         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
945         {
946             \exp_not:N \mode_leave_vertical:
947             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
948         }
949     }
950 }

```

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

## 12.18 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.

```

951 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
952 {
953     \keys_define:nn { enumext / #1 }
954     {
955         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
956         show-length .initial:n = false,
957     }
958 }
959 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `show-length`.)

## 12.19 Setting before, after and first keys

before

before\*

after

first

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

```

960 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
961 {
962     \keys_define:nn { enumext / #1 }
963     {
964         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
965         before .value_required:n = true,
966         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
967         before* .value_required:n = true,
968         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
969         after .value_required:n = true,
970         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
971         first .value_required:n = true,
972     }
973 }
974 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

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

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

```

975 \cs_new_protected:Nn \__enumext_before_args_exec:
976 {
977     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
978 }

```

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

```

979 \cs_new_protected:Nn \__enumext_before_keys_exec:
980 {
981     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
982 }

```

The function `\__enumext_after_stop_list:` executes the  $\{\langle code \rangle\}$  set by the `after` key “after” the `enumext` environment has finished:  $\backslash endlist \{\langle code \rangle\}$ .

```

983 \cs_new_protected:Nn \__enumext_after_stop_list:
984 {
985     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
986 }

```

The function `\__enumext_after_args_exec:` executes the  $\{\langle code \rangle\}$  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  $\backslash item$ :  $\backslash list \{\langle arg one \rangle\} \{\langle arg two \rangle\} \{\langle code \rangle\} \backslash item$ .

```

987 \cs_new_protected:Nn \__enumext_after_args_exec:
988 {
989     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
990 }

```

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

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

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
991 \cs_new_protected:Nn \__enumext_before_args_exec_v:
992 {
993     \tl_use:N \l__enumext_before_starred_key_v_tl
994 }
995 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
996 {
997     \tl_use:N \l__enumext_before_no_starred_key_v_tl
998 }
999 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1000 {
1001     \tl_use:N \l__enumext_after_stop_list_v_tl
1002 }
1003 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1004 {
1005     \tl_use:N \l__enumext_after_list_args_v_tl
1006 }

```

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

### 12.19.3 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:
1007 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1008 {
1009     \tl_use:N \l__enumext_before_starred_key_vii_tl
1010 }
1011 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1012 {
1013     \tl_use:N \l__enumext_before_starred_key_viii_tl
1014 }
1015 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1016 {
1017     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1018 }
1019 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1020 {

```

```

1021     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1022   }
1023   \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1024   {
1025     \tl_use:N \l__enumext_after_stop_list_vii_tl
1026   }
1027   \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1028   {
1029     \tl_use:N \l__enumext_after_stop_list_viii_tl
1030   }
1031   \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1032   {
1033     \tl_use:N \l__enumext_after_list_args_vii_tl
1034   }
1035   \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1036   {
1037     \tl_use:N \l__enumext_after_list_args_viii_tl
1038   }

```

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

## 12.20 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.

```

1039   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1040   {
1041     \keys_define:nn { enumext / #1 }
1042     {
1043       mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1044       mini-env .value_required:n = true,
1045       mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1046       mini-sep .initial:n = 0.3333em,
1047       mini-sep .value_required:n = true,
1048       columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1049       columns-sep .value_required:n = true,
1050       columns .int_set:c = { l__enumext_columns_#2_int },
1051       columns .initial:n = 1,
1052       columns .value_required:n = true,
1053     }
1054   }
1055   \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.

```

1056   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1057   {
1058     \keys_define:nn { enumext / #1 }
1059     {
1060       mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1061       mini-right .value_required:n = true,
1062       mini-right* .code:n = {
1063         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1064         \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1065       },
1066       mini-right* .value_required:n = true,
1067     }
1068   }
1069   \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

## 12.21 Adjustment of vertical spaces for multicols

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

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

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

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 of 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.

### 12.21.1 Adjustment of vertical spaces for multicols 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 `multicols` 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_multicols_above_X_skip` and `\l\_enumext_multicols_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

1070 \cs_new_protected:Npn \_enumext_undo_space:
1071 {
1072   \int_case:nnT { \lastnodetype }
1073   {
1074     { 11 }
1075     {
1076       \typeout{SKIIIIIIIIIIIIIP}
1077       \typeout{\the\lastskip}
1078       \unskip
1079     }
1080     { 12 }
1081     {
1082       \typeout{KERRRRRRRRRRRRRRRN}
1083       \typeout{\the\lastkern}
1084       \unkern
1085     }
1086     { 13 }
1087     {
1088       \typeout{PENALTYYYYYYYYYYYYYY}
1089       %%\unpenalty
1090     }
1091   }
1092 }
1093
1094
1095
1096 \cs_new_protected:Nn \_enumext_multi_set_vskip:
1097 {
1098   \skip_set:cn { l\_enumext_multicols_above_ \_enumext_level: _skip }
1099   {
1100     \skip_use:c { l\_enumext_topsep_ \_enumext_level: _skip }
1101   }
1102   \skip_set:cn { l\_enumext_multicols_below_ \_enumext_level: _skip }
1103   {
1104     \skip_use:c { l\_enumext_topsep_ \_enumext_level: _skip }
1105   }
1106   \_enumext_add_pre_parsep:
1107 }

```

(End of definition for `\_enumext_multi_set_vskip:`)

`\_enumext_add_pre_parsep:` The function `\_enumext_add_pre_parsep:` “*adjusted*” the value of `\l\_enumext_multicols_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*.

```

1108 \cs_new_protected:Nn \_enumext_add_pre_parsep:
1109 {
1110   \int_case:nn { \l\_enumext_level_int }

```

```

1111     {
1112         { 2 }{
1113             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1114             {
1115                 \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1116             }
1117         }
1118         { 3 }{
1119             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1120             {
1121                 \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1122             }
1123         }
1124         { 4 }{
1125             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1126             {
1127                 \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1128             }
1129         }
1130     }
1131 }

```

(End of definition for `\l__enumext_add_pre_parsep:`)

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

```

1132 \cs_new_protected:Nn \l__enumext_multi_addvspace:
1133 {
1134     \l__enumext_multi_set_vskip:
1135     \mode_if_vertical:T
1136     {
1137         \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1138         {
1139             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1140         }
1141         \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1142         {
1143             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1144         }
1145     }
1146     %%\l__enumext_undo_space:
1147     \par\nopagebreak
1148     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1149 }

```

(End of definition for `\l__enumext_multi_addvspace:`)

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

`\l__enumext_keyans_multi_set_vskip:` The function `\l__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`.

`\l__enumext_keyans_multi_addvspace:`

```

1150 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
1151 {
1152     \skip_set:Nn \l__enumext_multicols_above_v_skip
1153     {
1154         \l__enumext_topsep_v_skip
1155     }
1156     \skip_set:Nn \l__enumext_multicols_below_v_skip
1157     {
1158         \l__enumext_topsep_v_skip
1159     }
1160 }
1161 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
1162 {
1163     \l__enumext_keyans_multi_set_vskip:
1164     \mode_if_vertical:T
1165     {
1166         \skip_add:Nn \l__enumext_multicols_above_v_skip
1167         {
1168             \skip_use:N \l__enumext_partopsep_v_skip

```

```

1169     }
1170     \skip_add:Nn \l__enumext_multicols_below_v_skip
1171     {
1172         \skip_use:N \l__enumext_partopsep_v_skip
1173     }
1174 }
1175 \__enumext_undo_space:
1176 \par\nopagebreak
1177 \addvspace{ \l__enumext_multicols_above_v_skip }
1178 }

```

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

## 12.22 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.

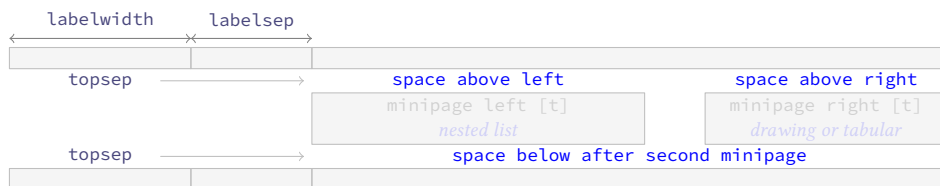


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

### 12.22.1 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_env*` 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`.

```

1179 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1180 {
1181     \skip_set:Nn \l__enumext_minipage_right_skip
1182     {
1183         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1184     }
1185     \mode_if_vertical:T
1186     {
1187         \skip_add:Nn \l__enumext_minipage_right_skip
1188         {
1189             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1190         }
1191     }
1192     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1193     %%\skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1194     \skip_set_eq:cN { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_
1195     \skip_set_eq:cN { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_
1196     %%\skip_if_eq:nnF { \l__enumext_minipage_after_skip } { \c_zero_skip }
1197     %% {
1198     \__enumext_previus_level_skip:
1199     %% }

```

Now we will see if the environment `multicols` is active, if so we set `\topskip=0pt` and then we make `\multicolsep` have the same value as `\l__enumext_minipage_right_skip`

```

1200 \int_compare:nNnTF
1201 { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }

```



```

1202     {
1203         \skip_zero:N \topskip
1204         \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \l__enumext_level: \skip }
1205         \skip_set:Nn \l__enumext_minipage_left_skip { 0.445\box_ht:N \strutbox }
1206     }
1207     {
1208         \skip_set:Nn \l__enumext_minipage_left_skip { 0.510\box_ht:N \strutbox }
1209     }
1210 }
1211 \cs_new_protected:Nn \l__enumext_previous_level_skip:
1212 {
1213     \int_case:nn { \l__enumext_level_int }
1214     {
1215         { 2 }{
1216             \skip_if_eq:nnTF { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1217             {
1218                 \typeout{SON-IGUALES}
1219                 \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1220                 \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1221             }
1222             {
1223                 \dim_compare:nNt { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after
1224                 {
1225                     \typeout{ITEM-SEP-MENOR}
1226                     \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1227                     \skip_sub:Nn \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_sk
1228                     \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1229                     \skip_add:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbo
1230                 }
1231                 \dim_compare:nNt { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after
1232                 {
1233                     \typeout{ITEM-SEP-MAY0000R}
1234                     \skip_set:Nn \l_tmpa_skip
1235                     {
1236                         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1237                     }
1238                     \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1239                     \skip_sub:Nn \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_sk
1240                     \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox + '
1241                     \skip_add:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbo
1242                 }
1243             }
1244         }
1245         { 3 }{
1246             \skip_if_eq:nnTF { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1247             {
1248                 \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1249             }
1250             {
1251                 \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1252             }
1253         }
1254         { 4 }{
1255             \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1256             {
1257                 \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1258             }
1259             {
1260                 \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1261             }
1262         }
1263     }
1264 }

```

The function `\l__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `\l__enumext_mini_env*` environment, taking into account whether  $\TeX$  is in  $\langle horizontal mode \rangle$  or  $\langle vertical mode \rangle$ . 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*.

```

1265 \cs_new_protected:Nn \l__enumext_minipage_add_space:
1266 {

```

```

1267 \__enumext_minipage_set_skip:
1268 \__enumext_undo_space:
1269 \mode_if_vertical:TF
1270 {
1271     \nopagebreak\nointerlineskip
1272 }
1273 {
1274     \par\nopagebreak\nointerlineskip
1275     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1276 }
1277 \addvspace{ \l__enumext_minipage_left_skip }
1278 }

```

(End of definition for \\_\_enumext\_minipage\_set\_skip: and \\_\_enumext\_minipage\_add\_space:.)

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

\\_\_enumext\_keyans\_minipage\_set\_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_env*` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

```

1279 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1280 {
1281     \skip_zero:N \l__enumext_minipage_after_skip
1282     \skip_zero:N \l__enumext_minipage_left_skip
1283     \skip_zero:N \l__enumext_minipage_right_skip
1284     \skip_set:Nn \l__enumext_minipage_right_skip
1285     {
1286         \l__enumext_topsep_v_skip
1287     }
1288     \mode_if_vertical:T
1289     {
1290         \skip_add:Nn \l__enumext_minipage_right_skip
1291         {
1292             \l__enumext_partopsep_v_skip
1293         }
1294     }
1295     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1296     %% prev level
1297     \skip_if_eq:nnF { \l__enumext_minipage_after_skip } { \c_zero_skip }
1298     {
1299         \skip_if_eq:nnTF { \l__enumext_itemsep_i_skip } { \c_zero_skip }
1300         {
1301             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1302         }
1303         {
1304             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1305         }
1306     }
1307     %% columns
1308     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1309     {
1310         \skip_zero:N \topskip
1311         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1312     }
1313 }

```

(End of definition for \\_\_enumext\_keyans\_minipage\_set\_skip:.)

\\_\_enumext\_keyans\_minipage\_add\_space:

The function \\_\_enumext\_keyans\_minipage\_add\_space: will apply the spaces set using `\addvspace` “above” the `\__enumext_mini_env*` environment in `keyans`, taking into account whether TeX is in *horizontal mode* or *vertical mode*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in `enumext`.

```

1314 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1315 {
1316     \__enumext_keyans_minipage_set_skip:
1317     \mode_if_vertical:TF
1318     {
1319         \nopagebreak\nointerlineskip
1320     }
1321     {

```

```

1322         \par\nopagebreak\nointerlineskip
1323         \skip_zero:N \l__enumext_partopsep_v_skip
1324     }
1325     \addvspace{ 0.245\box_ht:N \strutbox }
1326 }

```

(End of definition for \l\_\_enumext\_keyans\_minipage\_add\_space:.)

### 12.22.3 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_env*` environment in `enumext*` and `keyans*`.

```

1327 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1328 {
1329     \skip_zero_new:N \l__enumext_minipage_left_skip
1330     \skip_gzero_new:N \g__enumext_minipage_right_skip
1331     \skip_gzero_new:N \g__enumext_minipage_after_skip
1332     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1333     {
1334         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1335         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1336     }
1337     {
1338         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1339         \skip_gset:Nn \g__enumext_minipage_right_skip
1340         {
1341             \l__enumext_topsep_vii_skip
1342         }
1343         \skip_gset:Nn \g__enumext_minipage_after_skip
1344         {
1345             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1346         }
1347     }
1348 }
1349 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1350 {
1351     \skip_zero_new:N \l__enumext_minipage_after_skip
1352     \skip_zero_new:N \l__enumext_minipage_left_skip
1353     \skip_zero_new:N \l__enumext_minipage_right_skip
1354     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1355     {
1356         \skip_set:Nn \l__enumext_minipage_left_skip
1357         {
1358             0.5\box_dp:N \strutbox
1359         }
1360         \skip_set:Nn \l__enumext_minipage_right_skip
1361         {
1362             \l__enumext_partopsep_viii_skip
1363         }
1364         \skip_set:Nn \l__enumext_minipage_after_skip
1365         {
1366             1.6\box_dp:N \strutbox
1367         }
1368     }
1369     {
1370         \skip_set:Nn \l__enumext_minipage_left_skip
1371         {
1372             0.5875\box_dp:N \strutbox
1373         }
1374         \skip_set:Nn \l__enumext_minipage_right_skip
1375         {
1376             \l__enumext_topsep_viii_skip
1377         }
1378         \skip_set:Nn \l__enumext_minipage_after_skip
1379         {
1380             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1381         }
1382     }
1383 }

```

(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_env*` 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  $\text{\TeX}$  is in *(horizontal mode)* or *(vertical mode)*, since `\partopsep` is equal to `0pt` in both environments.

```

1384 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1385 {
1386   \__enumext_mini_set_vskip_vii:
1387   \par\nopagebreak
1388   \addvspace { \l__enumext_minipage_left_skip }
1389 }
1390 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1391 {
1392   \__enumext_mini_set_vskip_viii:
1393   \par\nopagebreak
1394   \addvspace { \l__enumext_minipage_left_skip }
1395 }

```

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

#### 12.22.4 The command `\miniright`

The command `\miniright` will close the `\__enumext_mini_env*` environment on the “left side”, open the `\__enumext_mini_env*` 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  $\text{\TeX}$  justification is maintained in the `\__enumext_mini_env*` 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.

```

1396 \NewDocumentCommand \miniright { s }
1397 {
1398   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1399   {
1400     \msg_error:nnn { enumext } { wrong-miniright-place }
1401   }
1402   % outside
1403   \bool_lazy_and:nnT
1404   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1405   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1406   {
1407     \msg_error:nnn { enumext } { wrong-miniright-place }
1408   }
1409   % starred env
1410   \bool_if:NT \l__enumext_starred_bool
1411   {
1412     \msg_error:nnn { enumext } { wrong-miniright-starred }
1413   }
1414   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1415   {
1416     \__enumext_keyans_mini_right_cmd:n {#1}
1417   }
1418   { \__enumext_mini_right_cmd:n {#1} }
1419 }

```

(End of definition for `\miniright`. This function is documented on page 10.)

`\__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 `\l__enumext_minipage_right_X_dim`, if so we close the `multicols` environment with the `\__enumext_mini_env*` environment on the “left side”, then we open the `\__enumext_mini_env*` 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.

```

1420 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1421 {
1422   \dim_compare:nNtF
1423   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1424   {
1425     \__enumext_multicols_stop:
1426     \int_compare:nNtF

```

```

1427         { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1428     {
1429         %%\skip_vertical:N \__enumext_minipage_after_skip
1430         %%\__enumext_undo_space: % remove previous
1431         \par\addvspace{ \__enumext_minipage_after_skip }
1432     }
1433 \end{__enumext_mini_env*}
1434 \hfill
1435 \begin{__enumext_mini_env*}
1436     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1437     % Add vertical space above
1438     \par\nointerlineskip
1439     \addvspace { \__enumext_minipage_right_skip }
1440     \bool_if:nF {#1}
1441     {
1442         \centering
1443     }
1444     \int_gzero:N \g__enumext_minipage_stat_int
1445 }
1446 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1447 % paranoia
1448 \RenewDocumentCommand \miniright { s }
1449 {
1450     \msg_error:nn { enumext } { many-miniright-used }
1451 }
1452 }

```

(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.

```

1453 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1454 {
1455     \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1456     {
1457         \__enumext_keyans_multicols_stop:
1458         \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1459         {
1460             \skip_vertical:N \__enumext_minipage_after_skip
1461         }
1462         \end{__enumext_mini_env*}
1463         \hfill
1464         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1465             % Add vertical space above
1466             \par\nointerlineskip
1467             \addvspace { \__enumext_minipage_right_skip }
1468             \bool_if:nF {#1}
1469             {
1470                 \centering
1471             }
1472             \int_gzero:N \g__enumext_minipage_stat_int
1473         }
1474         { \msg_error:nnn { enumext } { wrong-miniright-use } }
1475     % paranoia
1476     \RenewDocumentCommand \miniright { s }
1477     {
1478         \msg_error:nn { enumext } { many-miniright-used }
1479     }
1480 }

```

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

## 12.23 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.

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

```

above* 1481 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1482 {
below* 1483 \keys_define:nn { enumext / #1 }
1484 {
1485 above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1486 above .value_required:n = true,
1487 above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1488 \keys_set:nn { enumext / #1 } { above = {##1} },
1489 above* .value_required:n = true,
1490 below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1491 below .value_required:n = true,
1492 below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1493 \keys_set:nn { enumext / #1 } { below = {##1} },
1494 below* .value_required:n = true,
1495 }
1496 }
1497 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

### 12.23.1 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.

```

1498 \cs_new_protected:Nn \__enumext_vspace_above:
1499 {
1500 \skip_if_eq:nnF
1501 { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1502 {
1503 \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1504 {
1505 \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1506 }
1507 {
1508 \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1509 }
1510 }
1511 }

```

(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.

```

1512 \cs_new_protected:Nn \__enumext_vspace_below:
1513 {
1514 \skip_if_eq:nnF
1515 { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1516 {
1517 \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1518 {
1519 \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1520 }
1521 {
1522 \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1523 }
1524 }
1525 }

```

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

### 12.23.2 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.

```

1526 \cs_new_protected:Nn \__enumext_vspace_above_v:
1527 {
1528 \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1529 {
1530 \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1531 {
1532 \vspace*{ \l__enumext_vspace_above_v_skip }

```



```

1533     }
1534     { \vspace { \l__enumext_vspace_above_v_skip } }
1535   }
1536 }

```

(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.

```

1537 \cs_new_protected:Nn \__enumext_vspace_below_v:
1538 {
1539   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1540   {
1541     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1542     {
1543       \vspace*{ \l__enumext_vspace_below_v_skip }
1544     }
1545     { \vspace { \l__enumext_vspace_below_v_skip } }
1546   }
1547 }

```

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

### 12.23.3 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:

```

1548 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1549 {
1550   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1551   {
1552     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1553     {
1554       \vspace*{ \l__enumext_vspace_above_vii_skip }
1555     }
1556     { \vspace { \l__enumext_vspace_above_vii_skip } }
1557   }
1558 }
1559 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1560 {
1561   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1562   {
1563     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1564     {
1565       \vspace*{ \l__enumext_vspace_above_viii_skip }
1566     }
1567     { \vspace { \l__enumext_vspace_above_viii_skip } }
1568   }
1569 }

```

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

\\_\_enumext\_vspace\_below\_vii:

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.

\\_\_enumext\_vspace\_below\_viii:

```

1570 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1571 {
1572   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1573   {
1574     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1575     {
1576       \vspace*{ \l__enumext_vspace_below_vii_skip }
1577     }
1578     { \vspace { \l__enumext_vspace_below_vii_skip } }
1579   }
1580 }
1581 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1582 {
1583   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1584   {
1585     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1586     {
1587       \vspace*{ \l__enumext_vspace_below_viii_skip }

```

```

1588     }
1589     { \vspace { \l__enumext_vspace_below_viii_skip } }
1590   }
1591 }

```

(End of definition for `\__enumext_vspace_below_vii:` and `\__enumext_vspace_below_viii:`.)

## 12.24 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  $\langle keys \rangle$  passed to the optional argument of the “first level” of the environments `enumext` and `enumext*`, but, discarding some specific  $\langle keys \rangle$ . This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in [chat-Tex-SX](#)

We define the keys `series`, `resume` and `resume*` only for the “first level” of `enumext` and `enumext*`.

```

series
resume
resume*
1592 \cs_set_protected:Npn \__enumext_tmp:n #1
1593 {
1594   \keys_define:nn { enumext / #1 }
1595   {
1596     series .str_set:N = \__enumext_series_str,
1597     series .value_required:n = true,
1598     resume .code:n = \__enumext_resume_series:n {##1},
1599     resume* .code:n = \__enumext_resume_starred:,
1600     resume* .value_forbidden:n = true,
1601   }
1602 }
1603 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

### 12.24.1 Internal functions for series key

The function `\__enumext_filter_series:n` will be in charge of filtering the  $\langle keys \rangle$  we want to store where `{#1}` represents the optional value passed to the environment.

```

1604 \cs_new:Npn \__enumext_filter_series:n #1
1605 {
1606   \use:e
1607   {
1608     \keyval_parse:NNn
1609     \__enumext_filter_series_key:n
1610     \__enumext_filter_series_pair:nn {#1}
1611   }
1612 }

```

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

```

1613 \cs_new:Npn \__enumext_filter_series_key:n #1
1614 {
1615   \str_case:nnF {#1}
1616   {
1617     { resume } {} { resume* } {} { base-fix } {}
1618   }
1619   { , { \exp_not:n {#1} } }
1620 }

```

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

```

1621 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1622 {
1623   \str_case:nnF {#1}
1624   {
1625     { series } {} { resume } {} { start } {}
1626     { start* } {} { save-ans } {} { save-key } {}
1627   }
1628   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1629 }

```

(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 (§12.38) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§12.43).

```

1630 \cs_new_protected:Npn \__enumext_parse_series:n #1
1631 {
1632   \str_if_empty:NTF \l__enumext_series_str
1633   {
1634     \bool_if:NF \l__enumext_resume_active_bool
1635     {
1636       \__enumext_resume_last:n {#1}
1637     }
1638   }
1639   {
1640     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1641     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1642       { \__enumext_filter_series:n {#1} }
1643     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1644     {
1645       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1646     }
1647   }
1648 }

```

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. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the `series` key is not being used.

```

1649 \cs_new_protected:Npn \__enumext_resume_last:n #1
1650 {
1651   \bool_if:NT \l__enumext_standar_first_bool
1652   {
1653     \tl_gclear:N \g__enumext_standar_series_tl
1654     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1655   }
1656   \bool_if:NT \l__enumext_starred_first_bool
1657   {
1658     \tl_gclear:N \g__enumext_starred_series_tl
1659     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1660   }
1661 }

```

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

### 12.24.2 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_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 (§12.38) and the `enumext*` environment definition (§12.43).

```

1662 \cs_new_protected:Npn \__enumext_resume_save_counter:
1663 {
1664   \bool_if:NT \g__enumext_standar_bool
1665   {
1666     \tl_if_empty:NF \l__enumext_series_str
1667     {
1668       \int_gset_eq:cN

```

```

1669         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1670     }
1671 \tl_if_empty:NTF \l__enumext_resume_name_tl
1672 {
1673     \str_if_empty:NT \l__enumext_series_str
1674     {
1675         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1676     }
1677 }
1678 {
1679     \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1680     {
1681         \int_gset_eq:cN
1682         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1683     }
1684 }
1685 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1686 {
1687     \int_gset_eq:cN
1688     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1689 }
1690 }
1691 \bool_if:NT \g__enumext_starred_bool
1692 {
1693     \tl_if_empty:NF \l__enumext_series_str
1694     {
1695         \int_gset_eq:cN
1696         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1697     }
1698 \tl_if_empty:NTF \l__enumext_resume_name_tl
1699 {
1700     \str_if_empty:NT \l__enumext_series_str
1701     {
1702         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1703     }
1704 }
1705 {
1706     \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1707     {
1708         \int_gset_eq:cN
1709         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1710     }
1711 }
1712 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1713 {
1714     \int_gset_eq:cN
1715     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1716 }
1717 }
1718 }

```

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

### 12.24.3 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.

```

1719 \cs_new_protected:Npn \__enumext_resume_series:n #1
1720 {
1721     \tl_if_empty:NTF {#1}
1722     {
1723         \__enumext_resume_counter:n { }
1724     }
1725     {
1726         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1727         {
1728             \__enumext_resume_counter:n {#1}

```

```

1729         \bool_if:NT \g__enumext_standar_bool
1730         {
1731             \keys_set:nv { enumext / level-1 }
1732             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1733         }
1734         \bool_if:NT \g__enumext_starred_bool
1735         {
1736             \keys_set:nv { enumext / enumext* }
1737             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1738         }
1739     }
1740     {
1741         \bool_if:NT \g__enumext_standar_bool
1742         {
1743             \msg_error:nnn { enumext } { unknown-series } {#1}
1744         }
1745         \bool_if:NT \g__enumext_starred_bool
1746         {
1747             \msg_error:nnn { enumext } { unknown-series } {#1}
1748         }
1749     }
1750 }
1751 }

```

(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`.

```

1752 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1753 {
1754     \bool_set_true:N \l__enumext_resume_active_bool
1755     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1756     \tl_if_empty:NTF \l__enumext_resume_name_tl
1757     {
1758         \__enumext_resume_counter:
1759     }
1760     {
1761         \__enumext_resume_counter_series:
1762     }
1763     \__enumext_resume_counter_save_ans:
1764 }

```

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.

```

1765 \cs_new_protected:Nn \__enumext_resume_counter:
1766 {
1767     \bool_if:NT \g__enumext_standar_bool
1768     {
1769         \int_gincr:N \g__enumext_resume_int
1770         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1771     }
1772     \bool_if:NT \g__enumext_starred_bool
1773     {
1774         \int_gincr:N \g__enumext_resume_vii_int
1775         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1776     }
1777 }

```

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.

```

1778 \cs_new_protected:Nn \__enumext_resume_counter_series:
1779 {
1780     \bool_if:NT \g__enumext_standar_bool
1781     {
1782         \int_set:Nn \l__enumext_start_i_int

```

```

1783         {
1784             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1785         }
1786     }
1787     \bool_if:NT \g__enumext_starred_bool
1788     {
1789         \int_set:Nn \l__enumext_start_vii_int
1790         {
1791             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1792         }
1793     }
1794 }

```

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.

```

1795 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1796 {
1797     \bool_lazy_and:nnT
1798     { \bool_if_p:N \l__enumext_standar_first_bool }
1799     { \bool_if_p:N \l__enumext_store_active_bool }
1800     {
1801         \int_set:Nn \l__enumext_start_i_int
1802         {
1803             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1804         }
1805     }
1806     \bool_lazy_and:nnT
1807     { \bool_if_p:N \l__enumext_starred_first_bool }
1808     { \bool_if_p:N \l__enumext_store_active_bool }
1809     {
1810         \int_set:Nn \l__enumext_start_vii_int
1811         {
1812             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1813         }
1814     }
1815 }

```

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

#### 12.24.4 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.

```

1816 \cs_new_protected:Nn \__enumext_resume_starred:
1817 {
1818     \bool_if:NT \g__enumext_standar_bool
1819     {
1820         \tl_if_empty:NF \g__enumext_standar_series_tl
1821         {
1822             \__enumext_resume_counter:n { }
1823             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1824         }
1825     }
1826     \bool_if:NT \g__enumext_starred_bool
1827     {
1828         \tl_if_empty:NF \g__enumext_starred_series_tl
1829         {
1830             \__enumext_resume_counter:n { }
1831             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1832         }
1833     }
1834 }

```

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

### 12.25 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.

### 12.25.1 Setting save-ans key

`save-ans` We define the keys `save-ans` only for the “*first level*” of `enumext` and `enumext*`.

```

1835 \cs_set_protected:Npn \__enumext_tmp:n #1
1836 {
1837   \keys_define:nn { enumext / #1 }
1838   {
1839     save-ans .code:n = \__enumext_storing_set:n {##1},
1840     save-ans .value_required:n = true,
1841   }
1842 }
1843 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} }

```

(End of definition for `save-ans`.)

### 12.25.2 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:`.

`\__enumext_stop_save_ans_msg:`

```

1844 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1845 {
1846   \msg_term:nnVV { enumext } { save-ans-log }
1847   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1848 }
1849 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1850 {
1851   \msg_term:nnVV { enumext } { save-ans-log-hook }
1852   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1853 }

```

(End of definition for `\__enumext_start_save_ans_msg:` and `\__enumext_stop_save_ans_msg:`.)

`\__enumext_storing_set:n`  
`\__enumext_storing_exec:`

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.

```

1854 \cs_new_protected:Npn \__enumext_storing_set:n #1
1855 {
1856   \tl_set:Nx \l__enumext_store_name_tl {#1}
1857   \tl_if_empty:NTF \l__enumext_store_name_tl
1858   {
1859     \bool_lazy_or:nnT
1860     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1861     {
1862       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1863     }
1864   }
1865   {
1866     \bool_lazy_or:nnT
1867     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1868     {
1869       \__enumext_start_save_ans_msg:
1870       \__enumext_storing_exec:
1871     }
1872   }
1873 }

```

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 `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy {*⟨store name⟩*} into the global variable `\g__enumext_store_name_tl` and execute the function `\__enumext_anskey_env_make:V` creating the environment `anskey*` (§12.30). 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*`.

```

1874 \cs_new_protected:Nn \__enumext_storing_exec:
1875 {

```



```

1876 \bool_set_true:N \l__enumext_store_active_bool
1877 \bool_set_true:N \l__enumext_check_answers_bool
1878 \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1879 \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1880 \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1881 {
1882   \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1883   \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1884 }
1885 \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1886 {
1887   \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1888   \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1889 }
1890 \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1891 {
1892   \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1893   \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1894 }
1895 }

```

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

### 12.25.3 The check answer mechanism

The mechanism for checking that all questions are answered 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.

### 12.25.4 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

no-store

```

1896 \cs_set_protected:Npn \__enumext_tmp:n #1
1897 {
1898   \keys_define:nn { enumext / #1 }
1899   {
1900     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1901     check-ans .initial:n = false,
1902     check-ans .value_required:n = true,
1903     no-store .code:n = {
1904       \bool_set_false:N \l__enumext_check_answers_bool
1905       \bool_set_false:N \l__enumext_check_ans_key_bool
1906     },
1907     no-store .value_forbidden:n = true,
1908   }
1909 }

```

```

1910 \clist_map_inline:nn
1911 {
1912     level-1, level-2, level-3, level-4, enumext*
1913 }
1914 { \__enumext_tmp:n {#1} }

```

(End of definition for *check-ans* and *no-store*.)

### 12.25.5 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.

```

1915 \cs_new_protected:Nn \__enumext_check_ans_active:
1916 {
1917     \tl_if_empty:NF \l__enumext_store_name_tl
1918     {
1919         \bool_if:NT \l__enumext_check_answers_bool
1920         {
1921             \__enumext_check_ans_level:
1922         }
1923     }
1924 }

```

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*”.

```

1925 \cs_new_protected:Nn \__enumext_check_ans_level:
1926 {
1927     \int_case:nn { \l__enumext_level_int }
1928     {
1929         { 1 }{
1930             \bool_lazy_all:nT
1931             {
1932                 { \bool_if_p:N \g__enumext_starred_bool }
1933                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1934             }
1935             {
1936                 \int_gdecr:N \g__enumext_item_number_int
1937                 \bool_set_false:N \l__enumext_item_number_bool
1938             }
1939         }
1940         { 2 }{
1941             \int_gdecr:N \g__enumext_item_number_int
1942             \bool_set_false:N \l__enumext_item_number_bool
1943         }
1944         { 3 }{
1945             \int_gdecr:N \g__enumext_item_number_int
1946             \bool_set_false:N \l__enumext_item_number_bool
1947         }
1948         { 4 }{
1949             \int_gdecr:N \g__enumext_item_number_int
1950             \bool_set_false:N \l__enumext_item_number_bool
1951         }
1952     }

```

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.

```

1953 \int_case:nn { \l__enumext_level_h_int }
1954 {
1955     { 1 }{
1956         \bool_lazy_all:nT
1957         {
1958             { \bool_if_p:N \g__enumext_standar_bool }
1959             { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1960         }
1961         {
1962             \int_gdecr:N \g__enumext_item_number_int
1963             \bool_set_false:N \l__enumext_item_number_bool
1964         }

```

```

1965         }
1966     }
1967 }

```

(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.

```

1968 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1969 {
1970     \bool_lazy_and:nnT
1971     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1972     { \bool_if_p:N \g__enumext_standar_bool }
1973     {
1974         \bool_gset_true:N \g__enumext_check_ans_key_bool
1975     }
1976     \bool_lazy_and:nnT
1977     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1978     { \bool_if_p:N \g__enumext_starred_bool }
1979     {
1980         \bool_gset_true:N \g__enumext_check_ans_key_bool
1981     }
1982 }

```

(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:`.

```

1983 \cs_new_protected:Nn \__enumext_item_answer_diff:
1984 {
1985     \int_gset:Nn \g__enumext_item_answer_diff_int
1986     {
1987         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1988     }
1989 }

```

(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:`.

```

1990 \cs_new_protected:Nn \__enumext_check_ans_show:
1991 {
1992     \int_case:nn { \g__enumext_item_answer_diff_int }
1993     {
1994         { -1 } { \__enumext_check_ans_msg_less: }
1995         { 0 } { \__enumext_check_ans_msg_same_ok: }
1996         { 1 } { \__enumext_check_ans_msg_greater: }
1997     }
1998 }
1999 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2000 {
2001     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2002     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2003 }
2004 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2005 {
2006     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2007     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2008 }
2009 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2010 {
2011     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2012     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2013 }

```

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

`\__enumext_check_ans_log:` 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:`.

```

2014 \cs_new_protected:Nn \__enumext_check_ans_log:
2015 {
2016   \int_case:nn { \g__enumext_item_answer_diff_int }
2017   {
2018     { -1 } { \__enumext_check_ans_log_msg_less: }
2019     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2020     { 1 } { \__enumext_check_ans_log_msg_greater: }
2021   }
2022 }
2023 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2024 {
2025   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2026   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2027 }
2028 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2029 {
2030   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2031   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2032 }
2033 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2034 {
2035   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2036   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2037 }

```

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

### 12.25.6 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.

```

2038 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2039 {
2040   \int_compare:nNnT
2041   { \g__enumext_check_starred_cmd_int } = { 0 }
2042   {
2043     \msg_warning:nnnV
2044     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2045   }
2046   \int_compare:nNnT
2047   { \g__enumext_check_starred_cmd_int } > { 1 }
2048   {
2049     \msg_warning:nnnV
2050     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2051   }
2052   \int_gzero:N \g__enumext_check_starred_cmd_int
2053   \tl_clear:N \l__enumext_check_start_line_env_tl
2054 }

```

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

### 12.26 Keys and functions associated with storage

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*`.

```

2055 \cs_set_protected:Npn \__enumext_tmp:n #1
2056 {
2057   \keys_define:nn { enumext / #1 }
2058   {
2059     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2060     wrap-ans .initial:n =
2061       {
2062         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2063       },
2064     wrap-ans .value_required:n = true,
2065     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,

```

```

2066     wrap-opt .initial:n = [{##1}],
2067     wrap-opt .value_required:n = true,
2068     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2069     save-sep .initial:n = {, ~ },
2070     save-sep .value_required:n = true,
2071     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2072     mark-ans .initial:n = \textasteriskcentered,
2073     mark-ans .value_required:n = true,
2074     mark-pos .choice:,
2075     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2076     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2077     mark-pos / unknown .code:n =
2078         \msg_error:nnee { enumext } { unknown-choice }
2079         { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2080     mark-pos .initial:n = right,
2081     mark-pos .value_required:n = true,
2082     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2083     show-ans .initial:n = false,
2084     show-ans .value_required:n = true,
2085     show-pos .bool_set:N = \l__enumext_show_position_bool,
2086     show-pos .initial:n = false,
2087     show-pos .value_required:n = true,
2088     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2089     mark-ref .initial:n = \textasteriskcentered,
2090     mark-ref .value_required:n = true,
2091     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2092     save-ref .initial:n = false,
2093     save-ref .value_required:n = true,
2094 }
2095 }
2096 \clist_map_inline:nn { level-1, enumext* } { \l__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 2097 \cs_set_protected:Npn \l__enumext_tmp:n #1
show-ans 2098 {
show-pos 2099     \keys_define:nn { enumext / #1 }
2100     {
2101         mark-pos .choice:,
2102         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2103         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2104         mark-pos .initial:n = right,
2105         mark-pos .value_required:n = true,
2106         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2107         show-ans .initial:n = false,
2108         show-ans .value_required:n = true,
2109         show-pos .bool_set:N = \l__enumext_show_position_bool,
2110         show-pos .initial:n = false,
2111         show-pos .value_required:n = true,
2112     }
2113 }
2114 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

(End of definition for mark-pos, show-ans, and show-pos.)

### 12.26.1 Store optional arguments of the environments

The idea behind “*storing*” 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 arguments passed to the levels of the environment in which it is executed and “*storing*” them.

```

\__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 “*storing*” the *keys* filtered from the optional arguments 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*.

```

2115 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2116 {
2117     \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2118     {

```

```

2119         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2120         \tl_set:ce
2121             { l__enumext_store_save_key_ \__enumext_level: _tl }
2122             { \__enumext_filter_save_key:n {#1} }
2123     }
2124 }
2125 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2126 {
2127     \bool_if:NF \__enumext_store_save_key_vii_bool
2128     {
2129         \tl_clear:N \__enumext_store_save_key_vii_tl
2130         \tl_set:Nc \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2131     }
2132 }

```

(End of definition for \\_\_enumext\_store\_active\_keys:n and \\_\_enumext\_store\_active\_keys\_vii:n.)

### 12.26.2 Setting save-key key

Since this list structure will be stored in the *sequence* established by the `save-ans` key when executing `\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 list stored in the *sequence*.

`save-key` The values set by this key passed in the optional arguments of the `enumext` and `enumext*` environments will override the values of the `\__enumext_store_save_key_X_tl` variable set by the functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`.

Define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2133 \cs_set_protected:Npn \__enumext_tmp:n #1
2134 {
2135     \keys_define:nn { enumext / enumext* }
2136     {
2137         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2138         save-key .value_required:n = true,
2139     }
2140     \keys_define:nn { enumext / #1 }
2141     {
2142         save-key .code:n = \__enumext_parse_save_key:n {##1},
2143         save-key .value_required:n = true,
2144     }
2145 }
2146 \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 the filtered *keys* in the variable `\__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2147 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2148 {
2149     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2150     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2151     \tl_set:ce
2152         { l__enumext_store_save_key_ \__enumext_level: _tl }
2153         { \__enumext_filter_save_key:n {#1} }
2154 }
2155 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2156 {
2157     \bool_set_true:N \__enumext_store_save_key_vii_bool
2158     \tl_clear:N \__enumext_store_save_key_vii_tl
2159     \tl_set:Nc \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2160 }

```

(End of definition for \\_\_enumext\_parse\_save\_key:n and \\_\_enumext\_parse\_save\_key\_vii:n.)

### 12.26.3 Internal functions to store optional arguments

`\__enumext_filter_save_key:n` The function `\__enumext_filter_save_key:n` will be in charge of filtering the *keys* we want to store in *sequence* where `{#1}` represents the optional value passed to the environment.

```

\__enumext_filter_save_key:n
\__enumext_filter_save_key_key:n
\__enumext_filter_save_key_pair:nn
2161 \cs_new:Npn \__enumext_filter_save_key:n #1
2162 {
2163     \use:e
2164     {

```

```

2165         \keyval_parse:NNn
2166         \__enumext_filter_save_key_key:n
2167         \__enumext_filter_save_key_pair:nn {#1}
2168     }
2169 }

```

The function `\__enumext_filter_save_key_key:n` will be responsible for filtering the *⟨keys⟩* that are passed “without value” by excluding the `resume`, `resume*`, `no-store` and `base-fix` keys.

```

2170 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2171 {
2172     \str_case:nnF {#1}
2173     {
2174         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2175     }
2176     { , { \exp_not:n {#1} } }
2177 }

```

The function `\__enumext_filter_save_key_pair:nn` will be responsible for filtering the *⟨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.

```

2178 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2179 {
2180     \str_case:nnF {#1}
2181     {
2182         { series } {} { resume } {} { save-ans } {} { save-ref } {}
2183         { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2184         { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2185         { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2186         { mini-right* } {}
2187     }
2188     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2189 }

```

(End of definition for `\__enumext_filter_save_key:n`, `\__enumext_filter_save_key_key:n`, and `\__enumext_filter_save_key_pair:nn`.)

#### 12.26.4 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 content in *⟨prop list⟩* defined by `save-ans` key. The “stored content” is retrieved by means of the `\getkeyans` command.

The form in which the content 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.

```

2190 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2191 {
2192     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2193     {
2194         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2195     }
2196     { #1 }
2197 }
2198 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }

```

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

#### 12.26.5 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 content 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 content 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.

```

2199 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2200 {
2201     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2202 }
2203 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }

```

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



### 12.26.6 Functions for storing the list structure in the sequence

The memorization structure of the list is handled by the functions `\__enumext_store_level_open:` and `\__enumext_store_level_close:` which are executed per level within the `enumext` environment.

```

2204 \cs_new_protected:Nn \__enumext_store_level_open:
2205 {
2206   \bool_if:NT \l__enumext_check_answers_bool
2207   {
2208     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2209     {
2210       \__enumext_store_addto_seq:n
2211       {
2212         \item \begin{enumext}
2213       }
2214     }
2215     {
2216       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2217       {
2218         \item \begin{enumext} [
2219       }
2220       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2221       {
2222         ]
2223       }
2224       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2225     }
2226   }
2227 }
2228 \cs_new_protected:Nn \__enumext_store_level_close:
2229 {
2230   \bool_if:NT \l__enumext_check_answers_bool
2231   {
2232     \__enumext_store_addto_seq:n { \end{enumext} }
2233   }
2234 }

```

(End of definition for `\__enumext_store_level_open:` and `\__enumext_store_level_close:`.)

The memorization structure of the list is handled by the functions `\__enumext_store_level_open_vii:` and `\__enumext_store_level_close_vii:` which are executed in the `enumext*` environment.

```

2235 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2236 {
2237   \bool_if:NT \l__enumext_check_answers_bool
2238   {
2239     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2240     {
2241       \__enumext_store_addto_seq:n
2242       {
2243         \item \begin{enumext*}
2244       }
2245     }
2246     {
2247       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2248       {
2249         \item \begin{enumext*}[
2250       }
2251       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2252       {
2253         ]
2254       }
2255       \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2256     }
2257   }
2258 }
2259 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2260 {
2261   \bool_if:NT \l__enumext_check_answers_bool
2262   {
2263     \__enumext_store_addto_seq:n { \end{enumext*} }
2264   }
2265 }

```

(End of definition for `\__enumext_store_level_open_vii:` and `\__enumext_store_level_close_vii:`.)

### 12.26.7 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`

```
2266 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2267 {
2268   \mode_leave_vertical:
2269   \skip_horizontal:n { -\dim_use:N #2 }
2270   \makebox[0pt][ r ]
2271   {
2272     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2273     {
2274       \tl_use:N \l__enumext_mark_answer_sym_tl
2275     }
2276   }
2277   \skip_horizontal:n { \dim_use:N #2 }
2278 }
2279 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

(End of definition for `\__enumext_print_keyans_box:NN`.)

### 12.27 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 `⟨labels⟩`, we do not want to get double dots in our references, then we will place this in the variable `\l__enumext_newlabel_arg_two_tl`.

```
2280 \cs_new_protected:Nn \__enumext_store_internal_ref:
2281 {
2282   \cs_set_protected:Npn \__enumext_tmp:n ##1
2283   {
2284     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2285     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2286     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2287     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2288   }
2289   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2290   \cs_set:Npn \__enumext_tmp:n ##1
2291   { . \tl_use:c { \l__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.

```
2292   \bool_lazy_all:nT
2293   {
2294     { \bool_if_p:N \g__enumext_starred_bool }
2295     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2296   }
2297   {
2298     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2299     { \tl_use:N \l__enumext_label_copy_vii_tl }
2300   }
2301   \bool_lazy_all:nT
2302   {
2303     { \bool_not_p:n { \g__enumext_standar_bool } }
2304     { \bool_if_p:N \l__enumext_standar_bool }
2305     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2306   }
2307   {
2308     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2309     {
2310       \tl_use:N \l__enumext_label_copy_vii_tl
2311       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2312     }
2313   }
```

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.

```

2314 \bool_lazy_all:nT
2315 {
2316   { \bool_if_p:N \g__enumext_standar_bool }
2317   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2318   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2319 }
2320 {
2321   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2322   {
2323     \tl_use:N \l__enumext_label_copy_i_tl
2324     \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2325   }
2326 }
2327 \cs_set:Npn \l__enumext_tmp:n ##1
2328 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2329 \bool_lazy_all:nT
2330 {
2331   { \bool_if_p:N \g__enumext_standar_bool }
2332   { \bool_if_p:N \l__enumext_starred_bool }
2333   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2334 }
2335 {
2336   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2337   {
2338     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2339     \tl_use:N \l__enumext_label_copy_vii_tl
2340   }
2341 }

```

Now we set the variable `\l__enumext_newlabel_arg_one_tl` which will contain  $\langle \textit{store name} : \textit{position} \rangle$ .

```

2342 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2343 {
2344   \l__enumext_store_name_tl \c_colon_str
2345   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2346 }

```

Now execute the function `\l__enumext_newlabel:nn` and save the result in the variable `\l__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```

2347 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2348 {
2349   \l__enumext_newlabel:nn
2350   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2351   { \l__enumext_newlabel_arg_two_tl }
2352 }
2353 \l__enumext_write_aux_file_tl
2354 }

```

(End of definition for `\l__enumext_store_internal_ref:`)

## 12.28 Common functions for `\anskey` and `anskey*` environment

`\l__enumext_store_anskey_code:n`

The internal function `\l__enumext_store_anskey_code:n` first we pass the  $\langle \textit{argument} \rangle$  to the  $\langle \textit{prop list} \rangle$ , then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `save-ref` key and will call the function `\l__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”  $\langle \textit{argument} \rangle$ .

```

2355 \cs_new_protected:Npn \l__enumext_store_anskey_code:n #1
2356 {
2357   \int_gincr:N \g__enumext_item_anskey_int
2358   \l__enumext_store_addto_prop:n {#1}
2359   \bool_if:NT \l__enumext_store_ref_key_bool
2360   {
2361     \l__enumext_store_internal_ref:
2362   }
2363   \l__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the  $[\langle \textit{key} = \textit{val} \rangle]$  passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “store” in the  $\langle \textit{sequence} \rangle$ . 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`.

```

2364 \tl_clear:N \l__enumext_store_anskey_arg_tl

```

```

2365 \bool_lazy_and:nnT
2366 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2367 { \bool_not_p:n { \l__enumext_starred_bool } }
2368 {
2369   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2370 }
2371 \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 (*number*) to `\l__enumext_store_anskey_arg_tl`.

```

2372 \bool_lazy_and:nnT
2373 { \bool_not_p:n { \l__enumext_starred_bool } }
2374 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2375 {
2376   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2377   {
2378     ( \exp_not:V \l__enumext_store_item_join_int )
2379   }
2380 }

```

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 (*argument*) for `\anskey` or (*body*) for `anskey*`.

```

2381 \bool_if:NTF \l__enumext_store_item_star_bool
2382 {
2383   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2384   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2385   {
2386     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2387     {
2388       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2389     }
2390   }
2391   \dim_compare:nT
2392   {
2393     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2394   }
2395   {
2396     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2397     {
2398       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2399     }
2400   }
2401   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2402 }
2403 {
2404   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2405 }

```

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*).

```

2406 \bool_lazy_and:nnT
2407 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2408 { \bool_if_p:N \l__enumext_hyperref_bool }
2409 {
2410   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2411   {
2412     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2413     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2414   }
2415 }
2416 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2417 }

```

(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 (*argument*) passed to `\anskey` and the (*body*) for `anskey*` when using the `wrap-ans` key.

```

2418 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2419 {
2420   \par
2421   \bool_if:NTF \l__enumext_starred_bool
2422   {

```

```

2423     \__enumext_print_keyans_box:NN \__enumext_labelwidth_vii_dim \__enumext_labelsep_vii_d
2424   }
2425   {
2426     \__enumext_print_keyans_box:cc
2427     { \__enumext_labelwidth_ \__enumext_level: _dim }
2428     { \__enumext_labelsep_ \__enumext_level: _dim }
2429   }
2430   \__enumext_anskey_wrapper:n { #1 }
2431 }

```

(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 content stored in the `<prop list>` when using the `show-pos` key on the left margin next to the “*wraps*” `<argument>` passed to `\anskey` and the `<body>` in `anskey*` on the right side when using the `show-ans` key.

```

2432 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2433 {
2434   \bool_if:NT \__enumext_show_answer_bool
2435   {
2436     \__enumext_anskey_show_wrap_arg:n { #1 }
2437   }
2438   \bool_if:NT \__enumext_show_position_bool
2439   {
2440     \tl_set:Nx \__enumext_mark_answer_sym_tl
2441     {
2442       \group_begin:
2443       \exp_not:N \normalfont
2444       \exp_not:N \footnotesize [ \int_eval:n
2445       {
2446         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
2447       }
2448       ]
2449       \group_end:
2450     }
2451     \__enumext_anskey_show_wrap_arg:n { #1 }
2452   }
2453 }

```

(End of definition for \\_\_enumext\_anskey\_show\_wrap\_left:n.)

## 12.29 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>}`.

\\_\_enumext\_anskey\_unknown:n  
 \\_\_enumext\_anskey\_unknown:n

First we’ll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```

2454 \keys_define:nn { enumext / anskey }
2455 {
2456   break-col .bool_set:N = \__enumext_store_columns_break_bool,
2457   break-col .default:n = true,
2458   break-col .value_forbidden:n = true,
2459   item-join .int_set:N = \__enumext_store_item_join_int,
2460   item-join .value_required:n = true,
2461   item-star .bool_set:N = \__enumext_store_item_star_bool,
2462   item-star .default:n = true,
2463   item-star .value_forbidden:n = true,
2464   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
2465   item-sym* .value_required:n = true,
2466   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
2467   item-pos* .value_required:n = true,
2468   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2469 }

```

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`.

```

2470 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2471 {
2472   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2473 }

```

```

2474 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2475 {
2476   \tl_if_blank:nTF {#2}
2477   {
2478     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2479   }
2480   {
2481     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2482   }
2483 }

```

(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`.

```

2484 \NewDocumentCommand \anskey { o +m }
2485 {
2486   \__enumext_anskey_safe_outer:
2487   \group_begin:
2488     \bool_if:NT \l__enumext_check_answers_bool
2489     {
2490       \tl_if_novalue:nF {#1}
2491       {
2492         \keys_set:nn { enumext / anskey } {#1}
2493       }
2494       \tl_if_blank:nTF {#2}
2495       {
2496         \msg_error:nn { enumext } { anskey-empty-arg }
2497       }
2498       {
2499         \__enumext_anskey_safe_inner:
2500         \__enumext_store_anskey_code:n {#2}
2501       }
2502     }
2503   \group_end:
2504 }

```

(End of definition for `\anskey`. This function is documented on page 12.)

### 12.29.1 Internal functions for the command

`\__enumext_anskey_safe_outer:` 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.

`\__enumext_anskey_safe_inner:`

```

2505 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2506 {
2507   \bool_if:NF \l__enumext_store_active_bool
2508   {
2509     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2510   }
2511   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2512   {
2513     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2514   }
2515   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2516   {
2517     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2518   }
2519   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2520   {
2521     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2522   }
2523 }

```

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.

```

2524 \cs_new_protected:Nn \__enumext_anskey_safe_inner:

```

```

2525 {
2526   \int_incr:N \l__enumext_anskey_level_int
2527   \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2528   {
2529     \msg_error:nn { enumext } { anskey-nested }
2530   }
2531   \bool_if:NF \l__enumext_item_number_bool
2532   {
2533     \msg_error:nn { enumext } { anskey-unnumber-item }
2534   }
2535   \mode_if_math:T
2536   {
2537     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2538   }
2539 }

```

(End of definition for `\__enumext_anskey_safe_outer:` and `\__enumext_anskey_safe_inner:`)

### 12.30 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:` (§12.31) which is executed after the environment in which the key `save-ans` is active.

```

2540 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2541 {
2542   \cs_undefine:c { anskey* }
2543   \cs_undefine:c { endanskey* }
2544   \cs_undefine:c { __scontents_anskey*_env_begin: }
2545   \cs_undefine:c { __scontents_anskey*_env_end: }
2546 }

```

Detection of the `anskey*` environment outside the `enumext` and `enumext*` environments.

```

2547 \__enumext_before_env:nn { enumext }
2548 {
2549   \bool_lazy_and:nnT
2550   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2551   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2552   {
2553     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2554     {
2555       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2556     }
2557   }
2558 }
2559 \__enumext_before_env:nn { enumext* }
2560 {
2561   \bool_lazy_and:nnT
2562   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2563   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2564   {
2565     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2566     {
2567       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2568     }
2569   }
2570 }

```



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.

```

2571 \__enumext_before_env:nn { anskey* }
2572 {
2573   \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2574   {
2575     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2576   }
2577   \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2578   {
2579     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2580   }
2581   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2582   {
2583     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2584   }
2585   \bool_if:NF \__enumext_item_number_bool
2586   {
2587     \msg_error:nn { enumext } { anskey-unnumber-item }
2588   }
2589   \mode_if_math:T
2590   {
2591     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2592   }
2593 }

```

(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: ($12.25.1)` and we will execute it only if the variable `\__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`.

```

2594 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2595 {
2596   \bool_if:NT \__enumext_anskey_env_bool
2597   {
2598     \newenvsc{anskey*}[store-env=#1,print-env=false]
2599     \__enumext_anskey_env_exec:
2600   }
2601 }
2602 \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`.

```

2603 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2604 {
2605   \keys_define:nn { scontents / scontents }
2606   {
2607     break-col .bool_gset:N = \__enumext_store_columns_break_bool,
2608     break-col .default:n   = true,
2609     break-col .value_forbidden:n = true,
2610     item-join .int_gset:N   = \__enumext_store_item_join_int,
2611     item-join .value_required:n = true,
2612     item-star .bool_gset:N  = \__enumext_store_item_star_bool,
2613     item-star .default:n    = true,
2614     item-star .value_forbidden:n = true,
2615     item-sym* .tl_gset:N    = \__enumext_store_item_symbol_tl,
2616     item-sym* .value_required:n = true,
2617     item-pos* .dim_gset:N   = \__enumext_store_item_symbol_sep_dim,
2618     item-pos* .value_required:n = true,
2619     print-env .undefine:,
2620     store-env .undefine:,
2621     write-out .undefine:,
2622     unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2623   }
2624 }

```

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`.

```

2625 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2626 {
2627   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2628 }
2629 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2630 {
2631   \tl_if_blank:nTF {#2}
2632   {
2633     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2634   }
2635   {
2636     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2637   }
2638 }

```

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`.

```

2639 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2640 {
2641   \keys_define:nn { scontents / scontents }
2642   {
2643     break-col .undefine:,
2644     item-join .undefine:,
2645     item-star .undefine:,
2646     item-sym* .undefine:,
2647     item-pos* .undefine:,
2648     write-out .code:n = {
2649       \bool_set_false:N \l__scontents_storing_bool
2650       \bool_set_true:N \l__scontents_writing_bool
2651       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2652     },
2653     write-out .value_required:n = true,
2654     print-env .meta:nn = { scontents } { print-env = ##1 },
2655     print-env .default:n = true,
2656     store-env .meta:nn = { scontents } { store-env = ##1 },
2657     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2658   }
2659 }

```

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*.

```

2660 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2661 {
2662   \group_begin:
2663   \int_set:Nn \tex_newlinechar:D { ``^^J }
2664   \__scontents_rescan_tokens:x
2665   {
2666     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2667     #1
2668   }
2669 }

```

(End of definition for *anskey\** and others. This function is documented on page 13.)

`\__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⟩*.

```

2670 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2671 {
2672   \__enumext_before_env:nn { anskey* }
2673   {
2674     \__enumext_anskey_env_define_keys:
2675   }

```

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.

```

2676   \hook_if_empty:nF {env/anskey*/after}

```

```

2677     {
2678         \hook_gremove_code:nn {env/anskey*/after} { * }
2679     }
2680     \__enumext_after_env:nn { anskey* }
2681     {
2682         \__enumext_anskey_env_save_keys:
2683         \tl_clear:N \l__enumext_store_anskey_env_tl
2684         \tl_clear:N \l__enumext_store_anskey_opt_tl
2685         \bool_if:NT \l__enumext_check_answers_bool
2686         {
2687             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2688             {
2689                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2690             }
2691             \regex_match:nVTF
2692             { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2693             \l__enumext_store_anskey_env_tl
2694             {
2695                 \msg_error:nn { enumext } { anskey-empty-arg }
2696             }
2697             {
2698                 \__enumext_anskey_env_store:
2699             }
2700         }
2701         \__enumext_anskey_env_clean_vars:
2702         \__enumext_anskey_env_reset_keys:
2703     }
2704 }

```

• 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 `\l__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`.

```

2705 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2706 {
2707     \bool_lazy_and:nnT
2708     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2709     { \bool_not_p:n { \l__enumext_starred_bool } }
2710     {
2711         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2712     }

```

If the `item-join` key is present and the command is running under `enumext*` we will add to `\l__enumext_store_anskey_opt_tl`.

```

2713     \bool_lazy_and:nnT
2714     { \bool_not_p:n { \l__enumext_starred_bool } }
2715     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2716     {
2717         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2718         {
2719             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2720         }
2721     }

```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_opt_tl`.

```

2722     \bool_if:NT \g__enumext_store_item_star_bool
2723     {
2724         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2725         {
2726             ,item-star,
2727         }
2728         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2729         {
2730             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2731             {
2732                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,

```

```

2733         }
2734     }
2735     \dim_compare:nT
2736     {
2737         \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2738     }
2739     {
2740         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2741         {
2742             ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2743         }
2744     }
2745 }
2746 }

```

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`.

```

2747 \cs_new_protected:Nn \__enumext_anskey_env_store:
2748 {
2749     \group_begin:
2750     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2751     {
2752         \exp_args:Ne
2753         \__enumext_store_anskey_code:n
2754         {
2755             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2756         }
2757     }
2758     {
2759         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2760         \exp_args:Ne
2761         \__enumext_store_anskey_code:n
2762         {
2763             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2764         }
2765     }
2766     \group_end:
2767 }

```

The function `\__enumext_anskey_env_clean_vars:` will return the global variables used by the `<keys>` to their initial state.

```

2768 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2769 {
2770     \bool_gset_false:N \g__enumext_store_columns_break_bool
2771     \int_gzero:N \g__enumext_store_item_join_int
2772     \bool_gset_false:N \g__enumext_store_item_star_bool
2773     \tl_gclear:N \g__enumext_store_item_symbol_tl
2774     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2775 }

```

(End of definition for `\__enumext_anskey_env_save_keys:`, `\__enumext_anskey_env_store:`, and `\__enumext_anskey_env_clean_vars:`.)

### 12.31 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*` (§12.30) 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.

```

2776 \cs_new_protected:Nn \__enumext_execute_after_env:
2777 {
2778     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2779     {
2780         \tl_if_empty:NF \g__enumext_store_name_tl
2781         {
2782             \__enumext_stop_save_ans_msg:
2783             \__enumext_item_answer_diff:
2784             \__enumext_log_global_vars:

```

```

2785         \__enumext_log_answer_vars:
2786         \bool_if:NTF \g__enumext_check_ans_key_bool
2787         {
2788             \__enumext_check_ans_show:
2789         }
2790         { \__enumext_check_ans_log: }
2791         \__enumext_undefine_anskey_env:
2792     }
2793     \__enumext_reset_global_vars:
2794 }
2795 }

```

(End of definition for \\_\_enumext\_execute\_after\_env:.)

- This function is passed to the function \\_\_enumext\_after\_env:nn for the environments `enumext` (§12.38) and `enumext*` (§12.43) and it is executed only when the environments are not nested or at some level of these..

## 12.32 Common functions for keyans, keyans\* and keyanspic

### 12.32.1 Storing content in prop list

\\_\_enumext\_keyans\_addto\_prop:n

The function \\_\_enumext\_keyans\_addto\_prop:n will pass the contents of the current  $\langle label \rangle$  \l\_\_enumext\_label\_v\_tl for the `keyans` environment and the current  $\langle label \rangle$  \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, which will be passed to the  $\langle prop list \rangle$  defined by the `save-ans` key using the \\_\_enumext\_store\_addto\_prop:V.

```

2796 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2797 {
2798     \tl_clear:N \l__enumext_store_current_label_tl
2799     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2800     {
2801         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2802     }
2803     {
2804         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2805     }
2806     \tl_if_novalue:nF { #1 }
2807     {
2808         % Set save-sep
2809         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2810         {
2811             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2812         }
2813         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2814     }
2815     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2816 }

```

(End of definition for \\_\_enumext\_keyans\_addto\_prop:n.)

### 12.32.2 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 the `\anskey` command, basically because in this environments we are interested in the current  $\langle label \rangle$ . The mechanism defined here will allow to execute `\ref{⟨store name : position⟩}` and will return `1. (A)`.

\\_\_enumext\_keyans\_store\_ref:  
 \\_\_enumext\_keyans\_store\_ref\_aux\_i:  
 \\_\_enumext\_keyans\_store\_ref\_aux\_ii:

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  $\langle labels \rangle$  and remove the dots “.” from them, we do not want to get double dots in our references.

```

2817 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2818 {
2819     \bool_if:NT \l__enumext_store_ref_key_bool
2820     {
2821         \cs_set_protected:Npn \__enumext_tmp:n ##1
2822         {
2823             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2824             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2825             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2826             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2827         }
2828         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2829         \__enumext_keyans_store_ref_aux_i:

```

```

2830     }
2831 }

```

The auxiliary function `\__enumext_keyans_store_ref_aux_i:` set the variable `\l__enumext_newlabel_arg_one_tl` which will contain  $\langle \textit{store name} : \textit{position} \rangle$  analyzing whether the environment in which they are executed is `enumext*` or `enumext`.

```

2832 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2833 {
2834   \bool_if:NT \g__enumext_starred_bool
2835   {
2836     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2837   }
2838   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2839   {
2840     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2841     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2842   }
2843   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2844   {
2845     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2846     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2847   }
2848   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2849   {
2850     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2851     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2852   }
2853   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2854   {
2855     \l__enumext_store_name_tl \c_colon_str
2856     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2857   }
2858   \__enumext_keyans_store_ref_aux_ii:
2859 }

```

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.

```

2860 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2861 {
2862   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2863   {
2864     \l__enumext_newlabel:nn
2865     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2866     { \l__enumext_newlabel_arg_two_tl }
2867   }
2868   \l__enumext_write_aux_file_tl
2869 }

```

(End of definition for `\__enumext_keyans_store_ref:`, `\__enumext_keyans_store_ref_aux_i:`, and `\__enumext_keyans_store_ref_aux_ii:`.)

### 12.32.3 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  $\langle \textit{label} \rangle$  `\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  $\langle \textit{contents} \rangle$  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.

```

2870 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2871 {
2872   \tl_clear:N \l__enumext_store_current_label_tl
2873   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2874   {
2875     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2876   }
2877   {
2878     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2879   }
2880   \tl_if_no_value:nF { #1 }
2881   {
2882     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2883     {

```

```

2884         \tl_put_right:Ne \l__enumext_store_current_label_tl
2885         {
2886             \l__enumext_store_keyans_item_opt_sep_tl
2887         }
2888     }
2889     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2890 }
2891 \__enumext_keyans_addto_seq_link:
2892 }

```

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.

```

2893 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2894 {
2895     \bool_lazy_and:nnT
2896     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2897     { \bool_if_p:N \l__enumext_hyperref_bool }
2898     {
2899         \tl_put_right:Ne \l__enumext_store_current_label_tl
2900         {
2901             \hfill \exp_not:N \hyperlink
2902             {
2903                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2904             }
2905             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2906         }
2907     }
2908     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2909     \bool_if:NT \l__enumext_check_answers_bool
2910     {
2911         \int_gincr:N \g__enumext_item_anskey_int
2912     }
2913 }

```

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

#### 12.32.4 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.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:

```

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.

```

2914 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2915 {
2916     \tl_if_novalue:nF { #1 }
2917     {
2918         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2919     }
2920     \bool_if:NT \l__enumext_show_answer_bool
2921     {
2922         \__enumext_keyans_show_ans:
2923     }
2924     \bool_if:NT \l__enumext_show_position_bool
2925     {
2926         \__enumext_keyans_show_pos:
2927     }
2928 }
2929 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2930 {
2931     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2932     {
2933         \bool_lazy_or:nnT
2934         { \bool_if_p:N \l__enumext_show_answer_bool }
2935         { \bool_if_p:N \l__enumext_show_position_bool }
2936         {
2937             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2938         }

```



```

2939     }
2940   }
2941   \cs_new_protected:Nn \__enumext_keyans_show_ans:
2942   {
2943     \bool_if:NT \l__enumext_starred_bool
2944     {
2945       \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2946       \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2947     }
2948     \tl_put_left:Nn \l__enumext_label_v_tl
2949     {
2950       \__enumext_print_keyans_box:NN
2951       \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2952     }
2953   }
2954   \cs_new_protected:Nn \__enumext_keyans_show_pos:
2955   {
2956     \bool_if:NT \l__enumext_starred_bool
2957     {
2958       \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2959       \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2960     }
2961     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2962     {
2963       \tl_set:Ne \l__enumext_mark_answer_sym_tl
2964       {
2965         \group_begin:
2966         \exp_not:N \normalfont
2967         \exp_not:N \footnotesize [ \int_eval:n
2968           {
2969             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2970           }
2971         ]
2972         \group_end:
2973       }
2974     }
2975     {
2976       \tl_set:Ne \l__enumext_mark_answer_sym_tl
2977       {
2978         \group_begin:
2979         \exp_not:N \normalfont
2980         \exp_not:N \footnotesize [ \int_eval:n
2981           {
2982             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2983           }
2984         ]
2985         \group_end:
2986       }
2987     }
2988     \tl_put_left:Nn \l__enumext_label_v_tl
2989     {
2990       \__enumext_print_keyans_box:NN
2991       \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2992     }
2993   }

```

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

## 12.33 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.

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`.

```

2994 \cs_new_protected:Npn \__enumext_default_item:n #1
2995 {
2996   \tl_if_novalue:nTF {#1}
2997   {
2998     \bool_if:NT \__enumext_check_answers_bool
2999     {
3000       \int_gincr:N \g__enumext_item_number_int
3001       \bool_set_true:N \l__enumext_item_number_bool
3002     }
3003     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3004     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3005   }
3006   {
3007     \bool_set_eq:cc
3008     { l__enumext_wrap_label_ \__enumext_level: _bool }
3009     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3010     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3011   }
3012 }

```

(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`.

```

3013 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3014 {
3015   \tl_if_novalue:nTF {#1}
3016   {
3017     \tl_gset_eq:Nc
3018     \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3019   }
3020   {
3021     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3022   }
3023   \tl_if_novalue:nTF {#2}
3024   {
3025     \dim_set_eq:cc
3026     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3027     { l__enumext_labelsep_ \__enumext_level: _dim }
3028   }
3029   {
3030     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3031   }
3032   \bool_if:NT \__enumext_check_answers_bool
3033   {
3034     \int_gincr:N \g__enumext_item_number_int
3035     \bool_set_true:N \l__enumext_item_number_bool
3036   }
3037   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3038   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3039 }

```

The function `\__enumext_item_star_exec:` will be responsible for executing `\item*` for the `enumext` environment.

```

3040 \cs_new_protected:Nn \__enumext_item_star_exec:
3041 {
3042   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3043   {
3044     \mode_leave_vertical:

```

```

3045     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3046     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_tl }
3047     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3048   }
3049 }

```

(End of definition for \\_\_enumext\_starred\_item:nn and \\_\_enumext\_item\_star\_exec:.)

The function \\_\_enumext\_redefine\_item: will redefine the \item command in the enumext environment adding \item\*.

```

3050 \cs_new_protected:Nn \__enumext_redefine_item:
3051 {
3052   \RenewDocumentCommand \item { s o o }
3053   {
3054     \bool_if:nTF {##1}
3055     {
3056       \__enumext_starred_item:nn {##2} {##3}
3057     }
3058     { \__enumext_default_item:n {##2} }
3059   }
3060 }

```

The function \\_\_enumext\_make\_label: redefine \makeLabel for the keys align, font, wrap-label, wrap-label\* and \item\* for enumext environment.

```

3061 \cs_new_protected:Nn \__enumext_make_label:
3062 {
3063   \RenewDocumentCommand \makeLabel { m }
3064   {
3065     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3066     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3067     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3068     {
3069       \__enumext_item_star_exec:
3070       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3071     }
3072     { ##1 }
3073     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3074     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3075   }
3076 }

```

(End of definition for \\_\_enumext\_redefine\_item: and \\_\_enumext\_make\_label.)

These functions are passed to \\_\_enumext\_list\_arg\_two\_X: used in the definition of the enumext environment (§12.38).

### 12.34 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\*.

```

item-pos*
3077 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3078 {
3079   \keys_define:nn { enumext / #1 }
3080   {
3081     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3082     item-sym* .value_required:n = true,
3083     item-sym* .initial:n = {$\star$},
3084     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3085     item-pos* .value_required:n = true,
3086   }
3087 }
3088 \clist_map_inline:nn
3089 {
3090   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3091 }
3092 { \__enumext_tmp:nn #1 }

```

(End of definition for item-sym\* and item-pos\*.)

### 12.35 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).

### 12.35.1 Handling unknown keys for keyans and keyans\*

unknown

Define and set `unknown` key for `keyans` and `keyans*` environments.

```

3093 \cs_set_protected:Npn \__enumext_tmp:n #1
3094 {
3095   \keys_define:nn { enumext / #1 }
3096   {
3097     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3098   }
3099 }
3100 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3101 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3102 {
3103   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3104 }
3105 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3106 {
3107   \tl_if_blank:nTF {#2}
3108   {
3109     \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3110   }
3111   {
3112     \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3113   }
3114 }
```

(End of definition for `unknown`, `\__enumext_keyans_unknown_keys:n`, and `\__enumext_keyans_unknown_keys:nn`.)

### 12.35.2 Handling unknown keys for enumext\*

unknown

Define and set `unknown` key for `enumext*` environment.

```

3115 \keys_define:nn { enumext / enumext* }
3116 {
3117   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3118 }
```

Internal functions for handling `unknown` key.

```

3119 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3120 {
3121   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3122 }
3123 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3124 {
3125   \tl_if_blank:nTF {#2}
3126   {
3127     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3128   }
3129   {
3130     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3131   }
3132 }
```

(End of definition for `unknown`, `\__enumext_starred_unknown_keys:n`, and `\__enumext_starred_unknown_keys:nn`.)

### 12.35.3 Handling unknown keys for enumext

unknown

Defines and set the key `unknown` for `enumext` environment.

```

3133 \cs_set_protected:Npn \__enumext_tmp:n #1
3134 {
3135   \keys_define:nn { enumext / #1 }
3136   {
3137     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3138   }
3139 }
3140 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3141 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3142 {
3143   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3144 }
3145 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
```

```

3146 {
3147   \tl_if_blank:nTF {#2}
3148   {
3149     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3150   }
3151   {
3152     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3153   }
3154 }

```

(End of definition for `unknown`, `\__enumext_standar_unknown_keys:n`, and `\__enumext_standar_unknown_keys:nn`.)

## 12.36 Redefining `\item` and `\make_label` 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`.

```

3155 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3156 {
3157   \tl_if_novalue:nTF { #1 }
3158   {
3159     \bool_set_true:N \__enumext_wrap_label_v_bool
3160     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3161   }
3162   {
3163     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3164     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3165   }
3166 }

```

(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⟩`.

```

3167 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3168 {
3169   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3170   \__enumext_keyans_show_left:n { #1 }
3171   \bool_set_true:N \__enumext_wrap_label_v_bool
3172   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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 and finally *store* it in the `⟨sequence⟩`.

```

3173   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3174   \__enumext_keyans_addto_prop:n { #1 }
3175   \__enumext_keyans_store_ref:
3176   \__enumext_keyans_addto_seq:n { #1 }
3177   \int_gincr:N \g__enumext_check_starred_cmd_int
3178 }

```

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

`\item*`  
`\__enumext_keyans_redefine_item:`  
`\__enumext_keyans_make_label:`

The function `\__enumext_keyans_redefine_item:` is responsible for adding the *starred* 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.

```

3179 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3180 {
3181   \RenewDocumentCommand \item { s o }
3182   {
3183     \bool_if:nTF {##1}
3184     {
3185       \peek_remove_spaces:n
3186       {
3187         \__enumext_keyans_starred_item:n {##2}
3188       }
3189     }

```

```

3190     {
3191         \__enumext_keyans_default_item:n {##2}
3192     }
3193 }
3194 }

```

The function `\__enumext_keyans_make_label:` redefine `\makeLabel` for the keys `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `keyans` environment.

```

3195 \cs_new_protected:Nn \__enumext_keyans_make_label:
3196 {
3197     \RenewDocumentCommand \makeLabel { m }
3198     {
3199         \tl_use:N \l__enumext_label_fill_left_v_tl
3200         \tl_use:N \l__enumext_label_font_style_v_tl
3201         \bool_if:NTF \l__enumext_wrap_label_v_bool
3202         {
3203             \__enumext_wrapper_label_v:n { ##1 }
3204         }
3205         { ##1 }
3206         \tl_use:N \l__enumext_label_fill_right_v_tl
3207     }
3208 }

```

(End of definition for `\item*`, `\__enumext_keyans_redefine_item:`, and `\__enumext_keyans_make_label:`. This function is documented on page 14.)

- This functions are passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§12.37.2).

## 12.37 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`.

### 12.37.1 Calculation of `\leftmargin` and `\itemindent`

Consider the figure 9 where the default margins (on the left) of a list are represented.

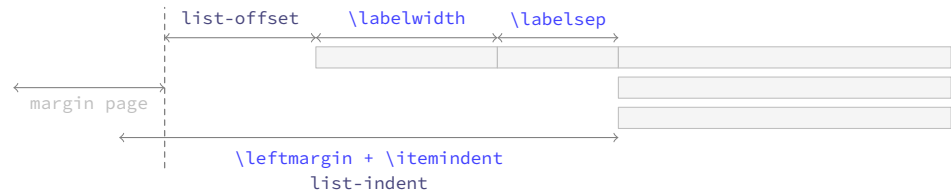


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

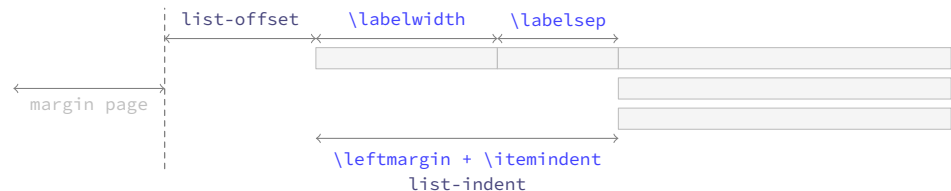


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

Where the default values will look like in the figure 11.

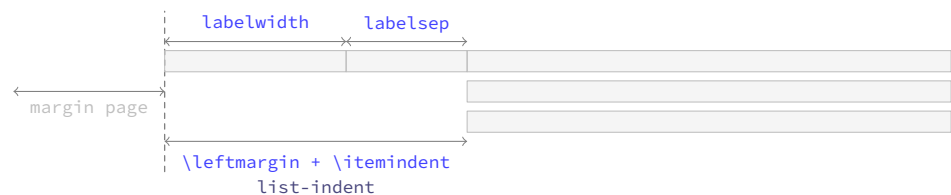


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

The function `\__enumext_calc_hspace:NNNNNNN` takes seven arguments to be able to determine horizontal spaces for all list environment:

```

#1: \l__enumext_labelwidth_X_dim      #2: \l__enumext_labelsep_X_dim
#3: \l__enumext_listoffset_X_dim      #4: \l__enumext_leftmargin_tmp_X_dim
#5: \l__enumext_leftmargin_X_dim      #6: \l__enumext_itemindent_X_dim
#7: \l__enumext_leftmargin_tmp_X_bool

```

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 (§12.37.2).

```

3209 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3210 {
3211   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3212   {
3213     \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3214     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3215   }
3216   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3217   {
3218     \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3219     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3220   }

```

If no value has been passed to the `labelwidth` and `labelsep` keys we set the default values for `\l__enumext_leftmargin_tmp_X_dim`.

```

3221   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

We now analyze the cases and set the values for `\leftmargin` and `\itemindent`.

```

3222   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3223   {
3224     \dim_set:Nn #6 { #1 + #2 - #4 }
3225     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3226   }
3227   {
3228     \dim_compare:nNnT { #4 } = { #1 + #2 }
3229     { \dim_set:Nn #6 { \c_zero_dim } }
3230     \dim_compare:nNnT { #4 } < { #1 + #2 }
3231     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3232     \dim_compare:nNnT { #4 } > { #1 + #2 }
3233     {
3234       \dim_set:Nn #6 { -#1 - #2 + #4 }
3235       \dim_set:Nn #6 { #6*-1 }
3236     }
3237     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3238   }
3239 }
3240 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(End of definition for `\__enumext_calc_hspace:NNNNNNN`.)

### 12.37.2 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:
3241 \cs_set_protected:Npn \__enumext_tmp:n #1
3242 {
3243   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3244   {
3245     \__enumext_calc_hspace:ccccc
3246     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3247     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3248     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3249     { \__enumext_leftmargin_tmp_#1_bool }
3250     \clist_map_inline:nn
3251     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3252     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3253     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3254     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3255     \usecounter { enumX#1 }
3256     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3257     \str_if_eq:nnTF {#1} { v }
3258     {
3259       \__enumext_keyans_redefine_item:

```



```

3260         \__enumext_keyans_make_label:
3261         \__enumext_keyans_ref:
3262         \__enumext_keyans_fake_item:
3263         \bool_if:cT { \__enumext_show_length_#1_bool }
3264         {
3265             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3266         }
3267     }
3268     {
3269         \__enumext_redefine_item:
3270         \__enumext_make_label:
3271         \__enumext_standar_ref:
3272         \__enumext_fake_item:
3273         \bool_if:cT { \__enumext_show_length_#1_bool }
3274         {
3275             \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_i
3276         }
3277     }
3278 }
3279 }
3280 \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 `\opt`. 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 `\parindent` to make it set the value of `\listparindent` and `\parsep` to set the value of `\parskip` locally.

```

3281 \cs_set_protected:Npn \__enumext_tmp:n #1
3282 {
3283     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3284     {
3285         \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3286         \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3287         \__enumext_calc_hspace:cccccc
3288         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3289         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3290         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3291         { \__enumext_leftmargin_tmp_#1_bool }
3292         \clist_map_inline:nn
3293         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3294         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3295         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3296         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3297         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3298         \skip_zero:N \partopsep
3299         \usecounter { enumX#1 }
3300         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3301         \__enumext_starred_ref:
3302         \str_if_eq:nnTF {#1} { vii }
3303         {
3304             \__enumext_fake_item_vii:
3305             \bool_if:cT { \__enumext_show_length_vii_bool }
3306             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3307         }
3308         {
3309             \__enumext_fake_item_viii:
3310             \bool_if:cT { \__enumext_show_length_#1_bool }
3311             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3312         }
3313     }
3314 }
3315 \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:.)

## 12.38 The environment enumext

`enumext` We create the `enumext` environment based on `list` environment by levels.

```

3316 \NewDocumentEnvironment{enumext}{0}{}
3317 {
3318     \__enumext_safe_exec:

```

```

3319     \__enumext_parse_keys:n {#1}
3320     \__enumext_before_list:
3321     \__enumext_start_store_level:
3322     \__enumext_start_list:nn
3323     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3324     {
3325         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3326         \__enumext_before_keys_exec:
3327     }
3328     \__enumext_set_item_width:
3329     \__enumext_after_args_exec:
3330 }
3331 {
3332     \__enumext_stop_list:
3333     \__enumext_stop_store_level:
3334     \__enumext_after_list:
3335 }

```

(End of definition for enumext. This function is documented on page 4.)

`\__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.

```

3336 \cs_new_protected:Nn \__enumext_set_item_width:
3337 {
3338     \dim_set:Nn \itemwidth
3339     {
3340         \linewidth
3341     }
3342     \dim_compare:nT
3343     {
3344         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3345     }
3346     {
3347         \dim_sub:Nn \itemwidth
3348         {
3349             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3350         }
3351     }
3352 }

```

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

`\__enumext_safe_exec:` The `\__enumext_safe_exec:` function first call the function `\__enumext_internal_mini_page:` to create the environment `__enumext_mini_env*`, then the function `\__enumext_is_not_nested:` which sets `\g__enumext_standar_bool` to “true” if we are not nested within `enumext*`, 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”.

```

3353 \cs_new_protected:Nn \__enumext_safe_exec:
3354 {
3355     \__enumext_internal_mini_page:
3356     \__enumext_is_not_nested:
3357     \int_incr:N \l__enumext_level_int
3358     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3359     { \msg_fatal:nn { enumext } { list-too-deep } }
3360     \bool_set_true:N \l__enumext_standar_bool
3361     \bool_set_false:N \l__enumext_starred_bool
3362     \__enumext_is_on_first_level:
3363 }

```

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

`\__enumext_parse_keys:n` The `\__enumext_parse_store_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 storage `(sequence)` if the key `save-key` is not active.

```

3364 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3365 {

```

```

3366     \tl_if_novalue:nF {#1}
3367     {
3368         \str_clear:N \__enumext_series_str
3369         \int_compare:nNnTF { \__enumext_level_int } = { 1 }
3370         {
3371             \keys_set:nn { enumext / level-1 } {#1}
3372             \__enumext_parse_series:n {#1}
3373             \__enumext_nested_base_line_fix:
3374         }
3375         {
3376             \exp_args:Ne \keys_set:nn
3377             { enumext / level-\int_use:N \__enumext_level_int } {#1}
3378         }
3379         \__enumext_store_active_keys:n {#1}
3380     }
3381 }

```

(End of definition for \\_\_enumext\_parse\_keys:n.)

\\_\_enumext\_start\_store\_level: The \\_\_enumext\_start\_store\_level: and \\_\_enumext\_stop\_store\_level: functions activate the level saving mechanism for storage in *(sequence)* for the command \anskey and the environment anskey\*.

\\_\_enumext\_stop\_store\_level:

```

3382 \cs_new_protected:Nn \__enumext_start_store_level:
3383 {
3384     \bool_lazy_all:nT
3385     {
3386         { \bool_if_p:N \__enumext_store_active_bool }
3387         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3388         { \bool_if_p:N \__enumext_standar_bool }
3389     }
3390     {
3391         \int_compare:nNnT { \__enumext_level_int } > { 1 }
3392         {
3393             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3394             \__enumext_store_level_open:
3395         }
3396     }

```

If enumext are nested in enumext\* add \\_\_enumext\_store\_level\_open: to preserve the stored structure.

```

3397     \bool_lazy_all:nT
3398     {
3399         { \bool_if_p:N \__enumext_store_active_bool }
3400         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3401         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
3402     }
3403     {
3404         \int_compare:nNnT { \__enumext_level_int } > { 0 }
3405         {
3406             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3407             \__enumext_store_level_open:
3408         }
3409     }
3410 }

```

Close the stored structure.

```

3411 \cs_new_protected:Nn \__enumext_stop_store_level:
3412 {
3413     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3414     {
3415         \__enumext_store_level_close:
3416     }
3417 }

```

(End of definition for \\_\_enumext\_start\_store\_level: and \\_\_enumext\_stop\_store\_level:.)

\\_\_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.

```

3418 \cs_new_protected:Nn \__enumext_before_list:
3419 {
3420     \__enumext_vspace_above:
3421     \__enumext_before_args_exec:
3422     \__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_env*` 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_env*` environment on the “left side”, always having a current `\linewidth` as *maximum width* between them.

```

3423 \dim_compare:nNt
3424 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3425 {
3426   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3427   {
3428     \linewidth
3429     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3430     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3431   }

```

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_env*` 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.

```

3432 \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3433 \int_gincr:N \g__enumext_minipage_stat_int
3434 \__enumext_minipage_add_space:
3435 \begin{__enumext_mini_env*}
3436   { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3437 }
3438 \__enumext_multicols_start:
3439 }

```

(End of definition for `\__enumext_before_list:`)

`\__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.

```

3440 \cs_new_protected:Nn \__enumext_multicols_start:
3441 {
3442   \int_compare:nNt
3443   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3444   {
3445     \dim_compare:nNt
3446     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3447     {
3448       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
3449       {
3450         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
3451         + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
3452         ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
3453         - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
3454       }
3455     }
3456     \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
3457     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3458     {
3459       \dim_zero:N \columnseprule
3460     }
3461   }

```

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.

```

3461 \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3462 {
3463   \skip_zero:N \multicolsep
3464   \__enumext_multi_addvspace:
3465 }
3466 \raggedcolumns
3467 \begin{multicols}{ \int_use:c { \l__enumext_columns_ \__enumext_level: _int } }
3468 }
3469 }

```

(End of definition for `\__enumext_multicols_start:`)

`\__enumext_multicols_stop:` The function `\__enumext_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_X_bool` is false (not nested in `__enumext_mini_env*`) we will apply our “vertical adjust” spacing.

```

3470 \cs_new_protected:Nn \__enumext_multicols_stop:
3471 {
3472   \int_compare:nNt
3473     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3474   {
3475     \end{multicols}
3476     \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3477       {
3478         \__enumext_undo_space:
3479         \__enumext_undo_space:
3480         \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \__enumext_level: _skip } }
3481       }
3482       {
3483         \__enumext_undo_space:
3484         \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \__enumext_level: _skip } }
3485       }
3486   }
3487 }

```

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

`\__enumext_after_list:` The function `\__enumext_after_list:` 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_env*` environment has not been closed), then close `__enumext_mini_env*` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3488 \cs_new_protected:Nn \__enumext_after_list:
3489 {
3490   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3491     {
3492       \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3493       {
3494         \msg_warning:nn { enumext } { missing-miniright }
3495         \miniright
3496       }
3497       \int_gzero:N \g__enumext_minipage_stat_int
3498       \__enumext_undo_space: % remove topsep + [partopsep]
3499       \end{__enumext_mini_env*}
3500     }
3501     {
3502       \__enumext_multicols_stop:
3503     }

```

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.

```

3504   \__enumext_after_stop_list:
3505   \__enumext_check_ans_key_hook:
3506   \__enumext_vspace_below:
3507   \bool_set_false:N \l__enumext_standar_bool
3508   \__enumext_resume_save_counter:
3509 }

```

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`.

```

3510 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

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

## 12.39 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”.

**keyans** Now we define the environment **keyans** also based on lists.

```

3511 \NewDocumentEnvironment{keyans}{0}{ }
3512 {
3513   \__enumext_keyans_safe_exec:
3514   \__enumext_keyans_parse_keys:n {#1}
3515   \__enumext_before_list_v:
3516   \__enumext_start_list:nn
3517   { \tl_use:N \__enumext_label_v_tl }
3518   {
3519     \__enumext_list_arg_two_v:
3520     \__enumext_before_keys_exec_v:
3521   }
3522   \__enumext_keyans_set_item_width:
3523   \__enumext_after_args_exec_v:
3524 }
3525 {
3526   \__enumext_check_starred_cmd:n { item }
3527   \__enumext_stop_list:
3528   \__enumext_after_list_v:
3529 }

```

(End of definition for **keyans**. This function is documented on page 14.)

**\\_\_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.

```

3530 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3531 {
3532   \dim_set:Nn \itemwidth
3533   {
3534     \linewidth
3535   }
3536   \dim_compare:nT
3537   {
3538     \l__enumext_listoffset_v_dim != \c_zero_dim
3539   }
3540   {
3541     \dim_sub:Nn \itemwidth
3542     {
3543       \l__enumext_listoffset_v_dim
3544     }
3545   }
3546 }

```

(End of definition for **\\_\_enumext\_keyans\_set\_item\_width:**.)

**\\_\_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.

```

3547 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3548 {
3549   \bool_if:NF \l__enumext_store_active_bool
3550   {
3551     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3552   }
3553   \int_incr:N \l__enumext_keyans_level_int
3554   \bool_set_true:N \l__enumext_keyans_env_bool
3555   \__enumext_keyans_name_and_start:
3556   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3557   \bool_set_false:N \l__enumext_store_active_bool
3558   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3559   {
3560     \msg_error:nn { enumext } { keyans-nested }
3561   }
3562   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3563   {
3564     \msg_error:nn { enumext } { keyans-wrong-level }
3565   }
3566 }

```

(End of definition for **\\_\_enumext\_keyans\_safe\_exec:**.)

`\__enumext_keyans_parse_keys:n`

Parse [*key* = *val*] for `keyans` environment.

```

3567 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3568 {
3569   \keys_set:nn { enumext / keyans } {#1}
3570 }
```

(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_after_list_v:
3571 \cs_new_protected:Nn \__enumext_before_list_v:
3572 {
3573   \__enumext_vspace_above_v:
3574   \__enumext_before_args_exec_v:
3575   \dim_compare:nNnT { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
3576   {
3577     \dim_set:Nn \__enumext_minipage_left_v_dim
3578     {
3579       \linewidth - \__enumext_minipage_right_v_dim - \__enumext_minipage_hsep_v_dim
3580     }
3581     \bool_set_true:N \__enumext_minipage_active_v_bool
3582     \int_gincr:N \g__enumext_minipage_stat_int
3583     \__enumext_keyans_minipage_add_space:
3584     \begin{__enumext_mini_env*}{ \__enumext_minipage_left_v_dim }
3585   }
3586   \__enumext_keyans_multicols_start:
3587 }
3588 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3589 {
3590   \int_compare:nNnT { \__enumext_columns_v_int } > { 1 }
3591   {
3592     \dim_compare:nNnT { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3593     {
3594       \dim_set:Nn \__enumext_columns_sep_v_dim
3595       {
3596         (
3597           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3598         ) / \__enumext_columns_v_int
3599         - \__enumext_listoffset_v_dim
3600       }
3601     }
3602     \dim_set_eq:NN \columnsep \__enumext_columns_sep_v_dim
3603     \dim_zero:N \columnseprule % no rule here
3604     \bool_if:NF \__enumext_minipage_active_v_bool
3605     {
3606       \skip_zero:N \multicolsep
3607       \__enumext_keyans_multi_addvspace:
3608     }
3609     \raggedcolumns
3610     \begin{multicols}{ \__enumext_columns_v_int }
3611   }
3612 }
3613 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3614 {
3615   \int_compare:nNnT { \__enumext_columns_v_int } > { 1 }
3616   {
3617     \end{multicols}
3618     \bool_if:NF \__enumext_minipage_active_v_bool
3619     {
3620       \par\addvspace{ \__enumext_multicols_below_v_skip }
3621     }
3622   }
3623 }
3624 \cs_new_protected:Nn \__enumext_after_list_v:
3625 {
3626   \bool_if:NTF \__enumext_minipage_active_v_bool
3627   {
3628     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3629     {
3630       \msg_warning:nn { enumext } { missing-miniright }
3631       \miniright
3632     }
3633   }
```



```

3633 \int_gzero:N \g__enumext_minipage_stat_int
3634 \unskip % remove topsep + [partopsep]
3635 \end{__enumext_mini_env*}
3636 \par\addvspace{ \l__enumext_minipage_after_skip }
3637 }
3638 {
3639 \__enumext_keyans_multicols_stop:
3640 }
3641 \bool_set_false:N \l__enumext_keyans_env_bool
3642 \__enumext_after_stop_list_v:
3643 \__enumext_vspace_below_v:
3644 }

```

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

## 12.40 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 and are placed inside `minipage` environments, with the `\label` underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to “adjust” the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.

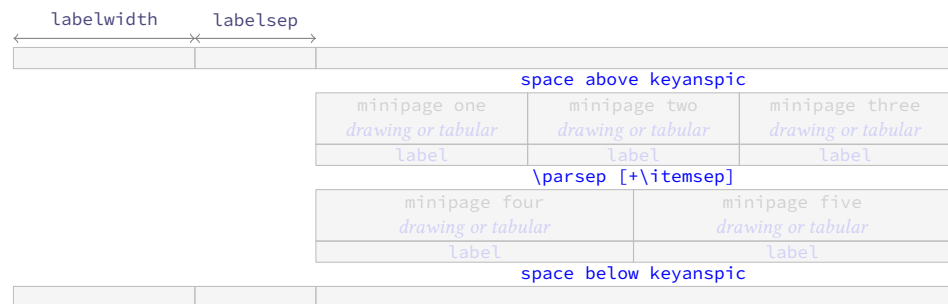


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

This implementation is adapted from the answer given by Enrico Gregorio in [How to process the body of an environment and divide it by a `\macro`?](#).

### 12.40.1 The command `\anspic`

`\anspic` The `\anspic` command take three arguments, the starred (\*) versions `\anspic*` and `\anspic*[\langle content \rangle]` store the current `\label` next to the `[\langle content \rangle]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```

3645 \NewDocumentCommand \anspic { s o +m }
3646 {

```

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error.

```

3647 \bool_if:NF \l__enumext_store_active_bool
3648 {
3649 \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3650 }
3651 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3652 {
3653 \msg_error:nn { enumext } { keyanspic-wrong-level }
3654 }
3655 \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3656 {
3657 \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3658 }

```

The three arguments are handled by the function `\__enumext_keyans_anspic_code:nnn` and stored in the sequence `\l__enumext_keyans_pic_body_seq` which is processed by the `keyanspic` environment.

```

3659 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3660 {
3661 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3662 }
3663 }

```

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

`\__enumext_keyans_anspic_code:nnn`

The function `\__enumext_keyans_anspic_code:nnn` will be in charge of handling the “counter” and  $\langle label \rangle$ , which will have the same configuration as the `keyans` environment.

```

3664 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3665 {
3666   \stepcounter { enumXvi }
3667   #3 \\\
3668   \bool_if:nT { #1 }
3669   {
3670     \__enumext_keyans_addto_prop:n { #2 }
3671     \__enumext_keyans_store_ref:
3672     \__enumext_keyans_addto_seq:n { #2 }
3673     \int_gincr:N \g__enumext_check_starred_cmd_int
3674     \bool_lazy_or:nnT
3675     { \bool_if_p:N \l__enumext_show_answer_bool }
3676     { \bool_if_p:N \l__enumext_show_position_bool }
3677     {
3678       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3679       \__enumext_keyans_show_left:n { #2 }
3680       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3681     }
3682   }
3683   \tl_use:N \l__enumext_label_font_style_v_tl
3684   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3685 }

```

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

#### 12.40.2 The environment `keyanspic`

`keyanspic`

Now we define the environment `keyanspic` based on list. The optional argument [ $\langle number\ above, number\ below \rangle$ ] will determine the number of `minipage` environments that will be above and below separated by `\parsep+\itemsep` within it.

```

3686 \NewDocumentEnvironment{keyanspic}{ o }
3687 {
3688   \__enumext_keyans_pic_safe_exec:
3689   \__enumext_start_list:nn
3690   { }
3691   {
3692     \__enumext_keyans_pic_arg_two:
3693   }

```

We apply the “adjusted” vertical spacing above the environment

```

3694   \vspace { \l__enumext_keyans_pic_above_skip }
3695 }

```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_keyans_pic_body_seq` and placed in `minipage` environments on a single line. Finally we check if `\anspic*` has been used, set the counter to zero and apply our “adjusted” vertical space below the environment.

```

3696 {
3697   \tl_if_novalue:nTF { #1 }
3698   {
3699     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3700   }
3701   { \__enumext_keyans_pic_do:n { #1 } }
3702   \__enumext_stop_list:
3703   \__enumext_check_starred_cmd:n { anspic }
3704   \setcounter { enumXvi } { 0 }
3705   \vspace { \l__enumext_topsep_v_skip }
3706   %\bool_set_false:N \l__enumext_store_active_bool
3707 }

```

(End of definition for `keyanspic`. This function is documented on page 15.)

`\__enumext_keyans_pic_safe_exec:`

The function `\__enumext_keyans_pic_safe_exec:` check nested and level position inside the `enumext` environment.

```

3708 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3709 {
3710   \int_incr:N \l__enumext_keyans_pic_level_int
3711   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3712   {
3713     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3714     }
3715     \__enumext_keyans_name_and_start:
3716 }

```

(End of definition for \\_\_enumext\_keyans\_pic\_safe\_exec:.)

\\_\_enumext\_keyans\_pic\_skip\_abs:N The function \\_\_enumext\_keyans\_pic\_skip\_abs:N will return a positive value \parsep.

```

3717 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3718 {
3719     \dim_compare:nNnT { #1 } < { 0pt }
3720     { \skip_set:Nn #1 { -#1 } }
3721 }

```

(End of definition for \\_\_enumext\_keyans\_pic\_skip\_abs:N.)

\\_\_enumext\_keyans\_pic\_arg\_two: The function \\_\_enumext\_keyans\_pic\_arg\_two: will be used in the second argument of the \\_\_enumext\_start\_list:nn function that defines the `keyanspic` environment, it will handle the setting of spaces.

```

3722 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3723 {

```

The first thing to do is to set the boolean variable \l\_\_enumext\_leftmargin\_tmp\_v\_bool handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```

3724     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3725     \__enumext_list_arg_two_v:

```

We will add the value of \itemsep to \parsep which we will use as vertical spacing between the above and below `minipage` environments. and adjust the value of \leftmargin, the label and counter are handled directly by the `anspic` command. Then we make equal to zero \labelwidth, \labelsep, \partopsep and \itemsep so that the horizontal and vertical spacing is not affected.

```

3726     \skip_add:Nn \parsep { \itemsep }
3727     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3728     \dim_zero:N \labelwidth
3729     \dim_zero:N \listparindent
3730     \dim_zero:N \labelsep
3731     \skip_zero:N \partopsep
3732     \skip_zero:N \itemsep

```

We set the value of \l\_\_enumext\_keyans\_pic\_above\_skip which we will use to apply our “adjust” space above `keyanspic`, finally we call \\_\_enumext\_item\_std:w followed by \scan\_stop: to prevent the error message returned by  $\TeX$  when not using the `\item` command.

```

3733     \__enumext_keyans_pic_skip_abs:N \parsep
3734     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3735     {
3736         \box_dp:N \strutbox
3737         + \l__enumext_topsep_v_skip
3738         - \parsep
3739     }
3740     \__enumext_item_std:w \scan_stop:
3741     % paranoia
3742     \RenewDocumentCommand \item {}
3743     {
3744         \msg_error:nn { enumext } { keyanspic-item-cmd }
3745     }
3746 }

```

(End of definition for \\_\_enumext\_keyans\_pic\_arg\_two:.)

\\_\_enumext\_keyans\_pic\_do:n The optional argument is split by comma and is handled directly by the function \\_\_enumext\_keyans\_pic\_do:n and passed to the function \\_\_enumext\_keyans\_pic\_row:n.

```

3747 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3748 {
3749     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3750 }
3751 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for \\_\_enumext\_keyans\_pic\_do:n.)

\\\_enumext\_keyans\_pic\_row:n

The function \\\_enumext\_keyans\_pic\_row:n will set the widths for the `minipage` environments and place the content `<stored>` by `\\anspic*` in the `\\_enumext_keyans_pic_body_seq` sequence inside them.

```

3752 \\cs_new_protected:Nn \\_enumext_keyans_pic_row:n
3753 {
3754   \\dim_set:Nn \\_enumext_keyans_pic_width_dim { \\linewidth / #1 }
3755   \\int_set:Nn \\_enumext_keyans_pic_above_int { \\_enumext_keyans_pic_below_int }
3756   \\int_set:Nn \\_enumext_keyans_pic_below_int { \\_enumext_keyans_pic_above_int + #1 }
3757   \\int_step_inline:nnn
3758     { \\_enumext_keyans_pic_above_int + 1 }
3759     { \\_enumext_keyans_pic_below_int }
3760   {
3761     \\_enumext_minipage:w [ b ]{ \\_enumext_keyans_pic_width_dim }
3762     \\centering
3763     \\seq_item:Nn \\_enumext_keyans_pic_body_seq { ##1 }
3764     \\_enumext_endminipage:
3765   }
3766   \\par
3767 }
```

(End of definition for \\\_enumext\_keyans\_pic\_row:n.)

## 12.41 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` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

## 12.42 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, it is necessary to redefine the command. This implementation is adapted from the answer given by Clea F. Rees (@cfr) in [footnotes in boxes compatible with hyperref](#).

```

3768 \\cs_new_protected:Nn \\_enumext_footnotetext:nn
3769 {
3770   \\footnotetext[#1]{#2}
3771 }
3772 \\cs_new_protected:Nn \\_enumext_renew_footnote:
3773 {
3774   \\seq_gclear:N \\g__enumext_footnote_arg_seq
3775   \\seq_gclear:N \\g__enumext_footnote_int_seq
3776   \\RenewDocumentCommand \\footnote { o +m }
3777   {
3778     \\tl_if_novalue:nTF {##1}
3779     {
3780       \\stepcounter{footnote}
3781       \\int_gset_eq:Nc \\g__enumext_footnote_int { c@footnote }
3782     }
3783     {
3784       \\int_gset:Nn \\g__enumext_footnote_int { ##1 }
3785     }
3786     \\footnotemark [ \\g__enumext_footnote_int ]
3787     \\seq_gput_right:Nn \\g__enumext_footnote_arg_seq { ##2 }
3788     \\seq_gput_right:NV \\g__enumext_footnote_int_seq \\g__enumext_footnote_int
3789   }
3790 }
3791 \\cs_new_protected:Nn \\_enumext_print_footnote:
3792 {
3793   \\seq_if_empty:NF \\g__enumext_footnote_int_seq
3794   {
3795     \\seq_map_pairwise_function:NNN
3796       \\g__enumext_footnote_int_seq
3797       \\g__enumext_footnote_arg_seq
3798       \\_enumext_footnotetext:nn
3799   }
3800 }
```

(End of definition for \\\_enumext\_footnotetext:nn, \\\_enumext\_renew\_footnote:, and \\\_enumext\_print\_footnote:.)

### 12.42.1 Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content 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 optional first argument (`\langle number \rangle`).

We set the default value for the *width of the box* containing the content of the items for `enumext*` environment.

```
\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:
```

```
3801 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3802 {
3803   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3804   {
3805     \dim_set:Nn \__enumext_columns_sep_vii_dim
3806     {
3807       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
3808       / \__enumext_columns_vii_int
3809     }
3810   }
3811   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
3812   \dim_set:Nn \__enumext_item_width_vii_dim
3813   {
3814     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
3815     / \__enumext_columns_vii_int
3816     - \__enumext_labelwidth_vii_dim
3817     - \__enumext_labelsep_vii_dim
3818   }
```

When the key `rightmargin` is active we must adjust the values.

```
3819   \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3820   {
3821     \dim_sub:Nn \__enumext_item_width_vii_dim
3822     {
3823       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
3824       / \__enumext_columns_vii_int
3825     }
3826     \dim_add:Nn \__enumext_columns_sep_vii_dim
3827     {
3828       \__enumext_rightmargin_vii_dim
3829     }
3830   }
```

Same implementation for the `keyans*` environment.

```
3832 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3833 {
3834   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3835   {
3836     \dim_set:Nn \__enumext_columns_sep_viii_dim
3837     {
3838       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
3839       / \__enumext_columns_viii_int
3840     }
3841   }
3842   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
3843   \dim_set:Nn \__enumext_item_width_viii_dim
3844   {
3845     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
3846     / \__enumext_columns_viii_int
3847     - \__enumext_labelwidth_viii_dim
3848     - \__enumext_labelsep_viii_dim
3849   }
3850   \dim_compare:nNnT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3851   {
3852     \dim_sub:Nn \__enumext_item_width_viii_dim
3853     {
3854       ( \__enumext_rightmargin_viii_dim * \__enumext_tmpa_viii_int )
3855       / \__enumext_columns_viii_int
3856     }
3857     \dim_add:Nn \__enumext_columns_sep_viii_dim
3858     {
3859       \__enumext_rightmargin_viii_dim
```

```

3860     }
3861   }
3862 }

```

(End of definition for `\__enumext_starred_columns_set_vii:` and `\__enumext_starred_columns_set_viii:`.)

### 12.42.2 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.

```

3863 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3864 {
3865   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3866   \int_compare:nNtT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3867   {
3868     \msg_warning:nnee { enumext } { item-joined }
3869     { \int_use:N \l__enumext_joined_item_vii_int }
3870     { \int_use:N \l__enumext_columns_vii_int }
3871     \int_set:Nn \l__enumext_joined_item_vii_int
3872     {
3873       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3874     }
3875   }
3876   \int_compare:nNtT
3877   { \l__enumext_joined_item_vii_int }
3878   >
3879   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3880   {
3881     \msg_warning:nnee { enumext } { item-joined-columns }
3882     { \int_use:N \l__enumext_joined_item_vii_int }
3883     {
3884       \int_eval:n
3885       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3886     }
3887     \int_set:Nn \l__enumext_joined_item_vii_int
3888     {
3889       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3890     }
3891   }
3892   \int_compare:nNtTF { \l__enumext_joined_item_vii_int } > { 1 }
3893   {
3894     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3895     \int_decr:N \l__enumext_joined_item_aux_vii_int
3896     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3897     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3898     \dim_set:Nn \l__enumext_joined_width_vii_dim
3899     {
3900       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3901       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3902         + \l__enumext_columns_sep_vii_dim
3903         ) * \l__enumext_joined_item_aux_vii_int
3904     }
3905     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3906   }
3907   {
3908     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3909     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3910   }
3911 }

```

Same implementation for the `keyans*` environment.

```

3912 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3913 {
3914   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3915   \int_compare:nNtT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3916   {
3917     \msg_warning:nnee { enumext } { item-joined }
3918     { \int_use:N \l__enumext_joined_item_viii_int }
3919     { \int_use:N \l__enumext_columns_viii_int }
3920     \int_set:Nn \l__enumext_joined_item_viii_int
3921     {

```

```

3922         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3923     }
3924 }
3925 \int_compare:nNt
3926 { \l__enumext_joined_item_viii_int }
3927 >
3928 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3929 {
3930     \msg_warning:nnee { enumext } { item-joined-columns }
3931     { \int_use:N \l__enumext_joined_item_viii_int }
3932     {
3933         \int_eval:n
3934         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3935     }
3936     \int_set:Nn \l__enumext_joined_item_viii_int
3937     {
3938         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3939     }
3940 }
3941 \int_compare:nNtTF { \l__enumext_joined_item_viii_int } > { 1 }
3942 {
3943     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3944     \int_decr:N \l__enumext_joined_item_aux_viii_int
3945     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3946     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3947     \dim_set:Nn \l__enumext_joined_width_viii_dim
3948     {
3949         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3950         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3951           + \l__enumext_columns_sep_viii_dim
3952         ) * \l__enumext_joined_item_aux_viii_int
3953     }
3954     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3955 }
3956 {
3957     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3958     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3959 }
3960 }

```

(End of definition for `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_viii:n`)

### 12.42.3 Functions for mini-env, mini-right and mini-right\* keys

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_env*` 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`.

```

3961 \cs_new_protected:Nn \__enumext_start_mini_vii:
3962 {
3963     \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3964     {
3965         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3966         {
3967             \linewidth
3968             - \l__enumext_minipage_right_vii_dim
3969             - \l__enumext_minipage_hsep_vii_dim
3970         }
3971         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3972         \dim_gset_eq:NN
3973         \g__enumext_minipage_right_vii_dim
3974         \l__enumext_minipage_right_vii_dim
3975         \__enumext_mini_addvspace_vii:
3976         \nointerlineskip\noindent
3977         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3978     }
3979 }

```

The function `\__enumext_stop_mini_vii:` closes the `\__enumext_mini_env*` environment on the left side, applies `\hfill` and sets the value of the variable `\g__enumext_minipage_active_vii_bool` to true which will be used in the function `\__enumext_after_env:nn` to execute the `\__enumext_mini_env*` on the “right side”.



```

3980 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3981 {
3982   \bool_if:NT \l__enumext_minipage_active_vii_bool
3983   {
3984     \end{__enumext_mini_env*}
3985     \hfill
3986     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3987   }
3988 }

```

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 `__enumext_mini_env*` 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.

```

3989 \__enumext_after_env:n {enumext*}
3990 {
3991   \bool_if:NT \g__enumext_minipage_active_vii_bool
3992   {
3993     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3994     \par\addvspace { \g__enumext_minipage_right_skip }
3995     \bool_if:NF \g__enumext_minipage_center_vii_bool
3996     {
3997       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3998       {
3999         \centering
4000       }
4001     }
4002     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4003     {
4004       \tl_use:N \g__enumext_miniright_code_vii_tl
4005     }
4006     \box_use_drop:N \l__enumext_miniright_code_vii_box
4007     \end{__enumext_mini_env*}
4008     \par\addvspace{ \g__enumext_minipage_after_skip }
4009   }
4010   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4011   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4012   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4013   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4014   \bool_gset_false:N \g__enumext_starred_bool
4015 }

```

(End of definition for `\__enumext_start_mini_vii:` and `\__enumext_stop_mini_vii:`)

`\__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.

```

4016 \cs_new_protected:Nn \__enumext_start_mini_viii:
4017 {
4018   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4019   {
4020     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4021     {
4022       \linewidth
4023       - \l__enumext_minipage_right_viii_dim
4024       - \l__enumext_minipage_hsep_viii_dim
4025     }
4026     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4027     \dim_gset_eq:NN
4028       \g__enumext_minipage_right_viii_dim
4029       \l__enumext_minipage_right_viii_dim
4030     \__enumext_mini_addvspace_viii:
4031     \nointerlineskip\noindent
4032     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
4033   }
4034 }
4035 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4036 {
4037   \bool_if:NT \l__enumext_minipage_active_viii_bool
4038   {
4039     \end{__enumext_mini_env*}
4040     \hfill

```

```

4041     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4042   }
4043 }
4044 \__enumext_after_env:nn {keyans*}
4045 {
4046   \bool_if:NT \g__enumext_minipage_active_viii_bool
4047   {
4048     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
4049     \par\addvspace { \g__enumext_minipage_right_skip }
4050     \bool_if:NF \g__enumext_minipage_center_viii_bool
4051     {
4052       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4053       {
4054         \centering
4055       }
4056     }
4057     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4058     {
4059       \tl_use:N \g__enumext_miniright_code_viii_tl
4060     }
4061     \box_use_drop:N \l__enumext_miniright_code_viii_box
4062     \end{__enumext_mini_env*}
4063     \par\addvspace{ \g__enumext_minipage_after_skip }
4064   }
4065   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4066   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4067   \tl_gclear:N \g__enumext_miniright_code_viii_tl
4068   \dim_gzero:N \g__enumext_minipage_right_viii_dim
4069 }

```

(End of definition for \\_\_enumext\_start\_mini\_viii: and \\_\_enumext\_stop\_mini\_viii:.)

## 12.43 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 \noindent and next to \item equal to \\_\_enumext\_start\_item\_tmp\_vii: which we will redefine later.

```

4070 \NewDocumentEnvironment{enumext*}{ o }
4071 {
4072   \__enumext_safe_exec_vii:
4073   \__enumext_parse_keys_vii:n {#1}
4074   \__enumext_before_list_vii:
4075   \__enumext_start_store_level_vii:
4076   \__enumext_start_list:nn { }
4077   {
4078     \__enumext_list_arg_two_vii:
4079     \__enumext_before_keys_exec_vii:
4080   }
4081   \__enumext_starred_columns_set_vii:
4082   \item[] \scan_stop:
4083   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4084   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4085 }
4086 {
4087   \__enumext_stop_item_tmp_vii:
4088   \__enumext_remove_extra_parsep_vii:
4089   \__enumext_stop_list:
4090   \__enumext_stop_store_level_vii:
4091   \__enumext_after_list_vii:
4092 }

```

(End of definition for enumext\*. This function is documented on page 4.)

\\_\_enumext\_safe\_exec\_vii: We will first call the function \\_\_enumext\_internal\_mini\_page: to create the environment **\_\_enumext-mini\_env\***, then the function \\_\_enumext\_is\_not\_nested: which sets \g\_\_enumext\_starred\_bool to true if we are not nested within **enumext**, 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.

```

4093 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4094 {

```

```

4095     \__enumext_internal_mini_page:
4096     \__enumext_is_not_nested:
4097     \int_incr:N \l__enumext_level_h_int
4098     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4099     {
4100         \msg_error:nn { enumext } { nested }
4101     }
4102     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4103     {
4104         \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4105     }
4106     \bool_set_true:N \l__enumext_starred_bool
4107     \bool_set_false:N \l__enumext_standar_bool
4108     \__enumext_is_on_first_level:
4109 }

```

(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 and finally we call the function \\_\_enumext\_nested\_base\_line\_fix: used by the key `base-fix`.

```

4110 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4111 {
4112     \tl_if_novalue:nF {#1}
4113     {
4114         \str_clear:N \l__enumext_series_str
4115         \keys_set:nn { enumext / enumext* } {#1}
4116         \__enumext_parse_series:n {#1}
4117         \__enumext_store_active_keys_vii:n {#1}
4118         \__enumext_nested_base_line_fix:
4119     }
4120 }

```

(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*`.

```

4121 \cs_new_protected:Nn \__enumext_before_list_vii:
4122 {
4123     \__enumext_vspace_above_vii:
4124     \__enumext_check_ans_active:
4125     \__enumext_before_args_exec_vii:
4126     \__enumext_start_mini_vii:
4127 }

```

(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: 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.

```

4128 \cs_new_protected:Nn \__enumext_after_list_vii:
4129 {
4130     \__enumext_stop_mini_vii:
4131     \__enumext_after_stop_list_vii:
4132     \__enumext_check_ans_key_hook:
4133     \__enumext_vspace_below_vii:
4134     \bool_set_false:N \l__enumext_starred_bool
4135     \__enumext_resume_save_counter:
4136 }

```

(End of definition for \\_\_enumext\_after\_list\_vii:.)

\\_\_enumext\_start\_store\_level\_vii:  
 \\_\_enumext\_stop\_store\_level\_vii:

The \\_\_enumext\_start\_store\_level\_vii: and \\_\_enumext\_stop\_store\_level\_vii: functions activate the level saving mechanism for storage in *(sequence)* of the \anskey command and anskey\* environment if enumext\* are nested in enumext.

```

4137 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4138 {
4139   \bool_if:NT \l__enumext_store_active_bool
4140   {
4141     \int_compare:nNt { \l__enumext_level_int } > { 0 }
4142     {
4143       \__enumext_store_level_open_vii:
4144     }
4145   }
4146 }
4147 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4148 {
4149   \bool_if:NT \l__enumext_store_active_bool
4150   {
4151     \int_compare:nNt { \l__enumext_level_int } > { 0 }
4152     {
4153       \__enumext_store_level_close_vii:
4154     }
4155   }
4156 }

```

(End of definition for \\_\_enumext\_start\_store\_level\_vii: and \\_\_enumext\_stop\_store\_level\_vii:.)

### 12.43.1 The command \item in enumext\*

\\_\_enumext\_start\_item\_tmp\_vii:

First we will call the function \\_\_enumext\_stop\_item\_tmp\_vii: that we will redefine later, we will increment the value of \l\_\_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.

```

4157 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4158 {
4159   \__enumext_stop_item_tmp_vii:
4160   \int_incr:N \l__enumext_item_column_pos_vii_int
4161   \int_gincr:N \g__enumext_item_count_all_vii_int
4162   \__enumext_item_peek_args_vii:
4163 }

```

(End of definition for \\_\_enumext\_start\_item\_tmp\_vii:.)

\\_\_enumext\_item\_peek\_args\_vii:

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).

```

4164 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4165 {
4166   \peek_meaning:NTF (
4167     { \__enumext_joined_item_vii:w }
4168     { \__enumext_joined_item_vii:w (1) }
4169   }

```

(End of definition for \\_\_enumext\_item\_peek\_args\_vii:.)

\\_\_enumext\_joined\_item\_vii:w

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.

```

4170 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4171 {
4172   \__enumext_starred_joined_item_vii:n {#1}
4173   \peek_meaning_remove:NTF *
4174     { \__enumext_starred_item_vii:w }
4175     { \__enumext_standar_item_vii:w }
4176 }

```

(End of definition for \\_\_enumext\_joined\_item\_vii:w.)

\\_\_enumext\_standar\_item\_vii:w

The function \\_\_enumext\_standar\_item\_vii:w will first look for the argument “[, if present it will set the state of the variable \l\_\_enumext\_wrap\_label\_opt\_vii\_bool equal to the state of the variable \l\_\_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 \l\_\_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 [ \l\_\_enumext\_label\_vii\_tl ].

```

4177 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4178 {
4179   \bool_set_false:N \l__enumext_item_starred_vii_bool
4180   \peek_meaning:NTF [
4181     {
4182       \bool_set_eq:NN
4183         \l__enumext_wrap_label_vii_bool
4184         \l__enumext_wrap_label_opt_vii_bool
4185       \__enumext_start_item_vii:w
4186     }
4187     {
4188       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4189       \legacy_if_set_true:n { @noitemarg }
4190       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4191     }
4192   }

```

(End of definition for \\_\_enumext\_standar\_item\_vii:w.)

\\_\_enumext\_starred\_item\_vii:w

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*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

\\_\_enumext\_starred\_item\_vii\_aux\_i:w

\\_\_enumext\_starred\_item\_vii\_aux\_ii:w

\\_\_enumext\_starred\_item\_vii\_aux\_iii:w

```

4193 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4194 {
4195   \bool_set_true:N \l__enumext_item_starred_vii_bool
4196   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4197   \peek_meaning:NTF [
4198     { \__enumext_starred_item_vii_aux_i:w }
4199     { \__enumext_starred_item_vii_aux_ii:w }
4200   }
4201   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4202   {
4203     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4204     \__enumext_starred_item_vii_aux_ii:w
4205   }
4206   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4207   {
4208     \peek_meaning:NTF [
4209       { \__enumext_starred_item_vii_aux_iii:w }
4210       {
4211         \dim_set_eq:NN
4212           \l__enumext_item_symbol_sep_vii_dim
4213           \l__enumext_labelsep_vii_dim
4214         \legacy_if_set_true:n { @noitemarg }
4215         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4216       }
4217     }
4218   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4219   {
4220     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4221     \legacy_if_set_true:n { @noitemarg }
4222     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4223   }

```

(End of definition for \\_\_enumext\_starred\_item\_vii:w and others.)

### 12.43.2 Real definition of \item in enumext\*

\\_\_enumext\_start\_item\_vii:w

The functions \\_\_enumext\_start\_item\_vii:w and \\_\_enumext\_stop\_item\_vii: executing the true definition of `\item` inside the `enumext*` environment. 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 and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is

given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

4224 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4225 {
4226   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4227   \legacy_if:nT { @noitemarg }
4228   {
4229     \legacy_if_set_false:n { @noitemarg }
4230     \legacy_if:nT { @nmbrlist }
4231     {
4232       \bool_if:NT \l__enumext_hyperref_bool
4233       {
4234         \legacy_if_set_true:n { @hyper@item }
4235       }
4236       \refstepcounter{enumXvii}
4237       \bool_if:NT \l__enumext_check_answers_bool
4238       {
4239         \int_gincr:N \g__enumext_item_number_int
4240         \bool_set_true:N \l__enumext_item_number_bool
4241       }
4242     }
4243   }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `\lrbox` environment. If the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the command `\footnote`, followed by printing the *symbol* defined for `\item*` if it is present and open a new group inside which we execute `font key` next to `\item` and the keys `wrap-label`, `wrap-label*`, `align`, close the group and execute the key `labelsep` and then the key `first`. Finally we open the `\minipage` environment and execute the `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```

4244   \group_begin:
4245   \lrbox{ \l__enumext_item_text_vii_box }
4246   \bool_if:NF \l__enumext_footnotes_key_bool
4247   {
4248     \__enumext_renew_footnote:
4249   }
4250   \bool_if:NT \l__enumext_item_starred_vii_bool
4251   {
4252     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4253     {
4254       \tl_gset_eq:NN
4255       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4256     }
4257     \mode_leave_vertical:
4258     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4259     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4260     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4261     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4262   }
4263   \group_begin:
4264   \tl_use:N \l__enumext_label_font_style_vii_tl
4265   \bool_if:NTF \l__enumext_wrap_label_vii_bool
4266   {
4267     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4268     { \__enumext_wrapper_label_vii:n {#1} }
4269   }
4270   {
4271     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4272   }
4273   \group_end:
4274   \skip_horizontal:N \l__enumext_labelsep_vii_dim
4275   \tl_use:N \l__enumext_after_list_args_vii_tl
4276   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4277   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4278   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4279   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4280 }

```

(End of definition for `\__enumext_start_item_vii:w`)

`\__enumext_stop_item_vii:` The function `\__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its *contents*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

4281 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4282 {
4283     \__enumext_endminipage:
4284     \endlrbox
4285     \group_end:
4286     \box_set_wd:Nn \l__enumext_item_text_vii_box
4287     {
4288         \l__enumext_joined_width_vii_dim
4289         + \l__enumext_labelwidth_vii_dim
4290         + \l__enumext_labelsep_vii_dim
4291     }
4292     \int_set:Nn \hbadness { 10000 }
4293     \box_use_drop:N \l__enumext_item_text_vii_box
4294     \bool_if:NF \l__enumext_footnotes_key_bool
4295     {
4296         \__enumext_print_footnote:
4297     }
4298     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4299     {
4300         \par\noindent
4301         \int_zero:N \l__enumext_item_column_pos_vii_int
4302     }
4303     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4304 }

```

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

`\__enumext_remove_extra_parsep_vii:` Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

4305 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4306 {
4307     \int_compare:nNnT
4308     {
4309         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4310     }
4311     =
4312     { 0 }
4313     {
4314         \par
4315         \vspace{ -\l__enumext_itemsep_vii_skip }
4316         \int_gzero:N \g__enumext_item_count_all_vii_int
4317     }
4318 }

```

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`.

```

4319 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

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

## 12.44 The environment `keyans*`

**keyans\*** First we will generate the environment and we will give a temporary definition to `\__enumext_stop_item_tmp_viii:` equal to `\noindent` and next to `\item` equal to `\__enumext_start_item_tmp_viii:` which we will redefine later.

```

4320 \NewDocumentEnvironment{keyans*}{ o }
4321 {
4322     \__enumext_safe_exec_viii:
4323     \__enumext_parse_keys_viii:n {#1}
4324     \__enumext_before_list_viii:
4325     \__enumext_start_list:nn { }
4326     {
4327         \__enumext_list_arg_two_viii:
4328         \__enumext_before_keys_exec_viii:
4329     }
4330     \__enumext_starred_columns_set_viii:
4331     \item[] \scan_stop:
4332     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent

```



```

4333     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4334   }
4335   {
4336     \__enumext_stop_item_tmp_viii:
4337     \__enumext_remove_extra_parsep_viii:
4338     \__enumext_check_starred_cmd:n { item }
4339     \__enumext_stop_list:
4340     \__enumext_after_list_viii:
4341   }

```

(End of definition for `keyans*`. This function is documented on page 14.)

`\__enumext_safe_exec_viii:` First check the maximum nesting level for the `keyans*` environment.

```

4342 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4343 {
4344   \int_incr:N \__enumext_keyans_level_h_int
4345   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4346   {
4347     \msg_error:nn { enumext } { nested }
4348   }
4349   \__enumext_keyans_name_and_start:
4350   \bool_if:NT \__enumext_starred_bool
4351   {
4352     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4353   }
4354   \bool_set_true:N \__enumext_starred_bool
4355   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4356   \bool_set_false:N \__enumext_store_active_bool
4357   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4358   {
4359     \msg_error:nn { enumext } { keyans-wrong-level }
4360   }
4361 }

```

(End of definition for `\__enumext_safe_exec_viii:`)

`\__enumext_parse_keys_viii:n` Parse [`key = val`] for `keyans*`.

```

4362 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4363 {
4364   \tl_if_novalue:nF {#1}
4365   {
4366     \keys_set:nn { enumext / keyans* } {#1}
4367   }
4368 }

```

(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`.

```

4369 \cs_new_protected:Nn \__enumext_before_list_viii:
4370 {
4371   \__enumext_vspace_above_viii:
4372   \__enumext_before_args_exec_viii:
4373   \__enumext_start_mini_viii:
4374 }

```

(End of definition for `\__enumext_before_list_viii:`)

`\__enumext_after_list_viii:` The function `\__enumext_after_list:` 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.

```

4375 \cs_new_protected:Nn \__enumext_after_list_viii:
4376 {
4377   \__enumext_stop_mini_viii:
4378   \__enumext_after_stop_list_viii:
4379   \__enumext_vspace_below_viii:
4380 }

```

(End of definition for `\__enumext_after_list_viii:`)

### 12.44.1 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 `\langle sequence \rangle` and `\langle prop list \rangle` 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_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. After that we will call the function `\_enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
4381 \cs_new_protected_nopar:Nn \_enumext_start_item_tmp_viii:
4382 {
4383   \_enumext_stop_item_tmp_viii:
4384   \int_incr:N \l\_enumext_item_column_pos_viii_int
4385   \int_gincr:N \g\_enumext_item_count_all_viii_int
4386   \_enumext_item_peek_args_viii:
4387 }
```

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

`\_enumext_item_peek_args_viii:`

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).

```
4388 \cs_new_protected:Nn \_enumext_item_peek_args_viii:
4389 {
4390   \peek_meaning:NTF (
4391     { \_enumext_joined_item_viii:w }
4392     { \_enumext_joined_item_viii:w (1) }
4393 }
```

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

`\_enumext_joined_item_viii:w`

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`.

```
4394 \cs_new_protected:Npn \_enumext_joined_item_viii:w (#1)
4395 {
4396   \_enumext_starred_joined_item_viii:n {#1}
4397   \peek_meaning_remove:NTF *
4398   { \_enumext_starred_item_viii:w }
4399   { \_enumext_standar_item_viii:w }
4400 }
```

(End of definition for `\_enumext_joined_item_viii:w`.)

`\_enumext_standar_item_viii:w`

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 [ \langle custom \rangle ]` 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 ]`.

```
4401 \cs_new_protected:Npn \_enumext_standar_item_viii:w
4402 {
4403   \bool_set_false:N \l\_enumext_item_starred_viii_bool
4404   \peek_meaning:NTF [
4405     {
4406       \bool_set_eq:NN
4407       \l\_enumext_wrap_label_viii_bool
4408       \l\_enumext_wrap_label_opt_viii_bool
4409       \_enumext_start_item_viii:w
4410     }
4411     {
4412       \bool_set_true:N \l\_enumext_wrap_label_viii_bool
4413       \legacy_if_set_true:n { @noitemarg }
4414       \_enumext_start_item_viii:w [ \l\_enumext_label_viii_tl ]
4415     }
4416 }
```

```

4415     }
4416 }

```

(End of definition for `\__enumext_standar_item_viii:w`.)

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w

```

The function `\__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[\langle content \rangle]`.

```

4417 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4418 {
4419   \bool_set_true:N \l__enumext_item_starred_viii_bool
4420   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4421   \peek_meaning:NTF [
4422     { \__enumext_starred_item_viii_aux_i:w }
4423     { \__enumext_starred_item_viii_aux_ii:w }
4424   }

```

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`.

```

4425 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4426 {
4427   \tl_clear:N \l__enumext_store_current_label_tl
4428   \tl_if_no_value:nF { #1 }
4429   {
4430     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4431     {
4432       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4433       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4434     }
4435     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4436   }
4437   \__enumext_starred_item_viii_aux_ii:w
4438 }
4439 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4440 {
4441   \legacy_if_set_true:n { @noitemarg }
4442   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4443 }

```

(End of definition for `\__enumext_starred_item_viii:w`, `\__enumext_starred_item_viii_aux_i:w`, and `\__enumext_starred_item_viii_aux_ii:w`.)

```
\__enumext_starred_item_exec:
```

The function `\__enumext_starred_item_exec:` will be in charge of storing the current `\label` for `\item*` followed by the `[\langle content \rangle]` for `\item*[\langle content \rangle]` 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.

```

4444 \cs_new_protected:Nn \__enumext_starred_item_exec:
4445 {
4446   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4447   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4448   \__enumext_keyans_store_ref:
4449   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4450   \__enumext_keyans_addto_seq_link:
4451   \int_gincr:N \g__enumext_check_starred_cmd_int
4452   \bool_if:NT \l__enumext_show_answer_bool
4453   {
4454     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4455   }
4456   \bool_if:NT \l__enumext_show_position_bool
4457   {
4458     \tl_set:Ne \l__enumext_mark_answer_sym_tl
4459     {
4460       \group_begin:
4461       \exp_not:N \normalfont
4462       \exp_not:N \footnotesize [ \int_eval:n
4463         {
4464           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4465         }
4466       ]
4467       \group_end:
4468     }

```

```

4469         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4470     }
4471 }

```

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

#### 12.44.2 Real definition of `\item` in keyans\*

The implementation at this point is very similar to that of the `enumext*` environment.

```

4472 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4473 {
4474     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4475     \legacy_if:nT { @noitemarg }
4476     {
4477         \legacy_if_set_false:n { @noitemarg }
4478         \legacy_if:nT { @nmbrrlist }
4479         {
4480             \bool_if:NT \l__enumext_hyperref_bool
4481             {
4482                 \legacy_if_set_true:n { @hyper@item }
4483             }
4484             \refstepcounter{enumXviii}
4485         }
4486     }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment.

```

4487     \group_begin:
4488     \lrbox{ \l__enumext_item_text_viii_box }
4489     \bool_if:NF \l__enumext_footnotes_key_bool
4490     {
4491         \__enumext_renew_footnote:
4492     }
4493     \bool_if:NT \l__enumext_item_starred_viii_bool
4494     {
4495         \__enumext_starred_item_exec:
4496     }
4497     \group_begin:
4498     \tl_use:N \l__enumext_label_font_style_viii_tl
4499     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4500     {
4501         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4502         { \__enumext_wrapper_label_viii:n {#1} }
4503     }
4504     {
4505         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4506     }
4507     \group_end:
4508     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4509     \tl_use:N \l__enumext_after_list_args_viii_tl
4510     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4511     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4512     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4513     \bool_if:NT \l__enumext_item_starred_viii_bool
4514     {
4515         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4516         \__enumext_keyans_show_item_opt:
4517         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim
4518     }
4519     {
4520         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4521     }
4522 }

```

(End of definition for `\__enumext_start_item_viii:w`.)

`\__enumext_stop_item_viii:` The function `\__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its *contents*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

4523 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4524 {
4525     \__enumext_endminipage:
4526     \endlrbox

```

```

4527 \group_end:
4528 \box_set_wd:Nn \l__enumext_item_text_viii_box
4529 {
4530   \l__enumext_joined_width_viii_dim
4531   + \l__enumext_labelwidth_viii_dim
4532   + \l__enumext_labelsep_viii_dim
4533 }
4534 \int_set:Nn \hbadness { 10000 }
4535 \box_use_drop:N \l__enumext_item_text_viii_box
4536 \bool_if:NF \l__enumext_footnotes_key_bool
4537 {
4538   \l__enumext_print_footnote:
4539 }
4540 \int_compare:nNnTF
4541 { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4542 {
4543   \par\noindent
4544   \int_zero:N \l__enumext_item_column_pos_viii_int
4545 }
4546 { \hspace{ \l__enumext_columns_sep_viii_dim } }
4547 }

```

(End of definition for `\l__enumext_stop_item_viii:`)

`\l__enumext_remove_extra_parsep_viii:`

Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

4548 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_viii:
4549 {
4550   \int_compare:nNnT
4551   {
4552     \int_mod:nn
4553     { \g__enumext_item_count_all_viii_int }
4554     { \l__enumext_columns_viii_int }
4555   }
4556   =
4557   { 0 }
4558   {
4559     \par
4560     \vspace{ -\l__enumext_itemsep_viii_skip }
4561     \int_gzero:N \g__enumext_item_count_all_viii_int
4562   }
4563 }

```

(End of definition for `\l__enumext_remove_extra_parsep_viii:`)

## 12.45 The command `\getkeyans`

`\getkeyans`

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*` from  $\langle \textit{prop list} \rangle$  defined by `save-ans` key.

```

4564 \NewDocumentCommand \getkeyans { m }
4565 {
4566   \exp_args:Ne \l__enumext_getkeyans_aux:n
4567   { \tl_to_str:e { \text_expand:n {#1} } }
4568 }

```

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

`\l__enumext_getkeyans_aux:n`

The internal function `\l__enumext_getkeyans_aux:n` is in charge of *splitting* the  $\langle \textit{argument} \rangle$  using “.”. If “.” is omitted it will return an error.

```

4569 \cs_new_protected:Npn \l__enumext_getkeyans_aux:n #1
4570 {
4571   \str_if_in:nnTF {#1} { : }
4572   {
4573     \use:e
4574     {
4575       \cs_set:Npn \exp_not:N \l__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4576       { {##1} {##2} }
4577     }
4578     \exp_after:wN \l__enumext_getkeyans:nn \l__enumext_tmp:w #1 \scan_stop:
4579   }
4580   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4581 }

```

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

`\__enumext_getkeyans:nn` 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*.

```

4582 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4583 {
4584   \prop_if_exist:cTF { g__enumext_#1_prop }
4585   {
4586     \prop_item:cn { g__enumext_#1_prop }{#2}
4587   }
4588   {
4589     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4590   }
4591 }

```

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

## 12.46 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]`.

```

4592 \keys_define:nn { enumext / print }
4593 {
4594   print* .code:n      = \keys_precompile:neN { enumext / enumext* }
4595                     { \__enumext_filter_save_key:n {#1} }
4596                     \l__enumext_print_keyans_starred_tl, % starred cmd
4597   print* .initial:n   = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4598   print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4599                     { \__enumext_filter_save_key:n {#1} }
4600                     \l__enumext_print_keyans_i_tl,
4601   print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4602   print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4603                     { \__enumext_filter_save_key:n {#1} }
4604                     \l__enumext_print_keyans_ii_tl,
4605   print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4606   print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4607                     { \__enumext_filter_save_key:n {#1} }
4608                     \l__enumext_print_keyans_iii_tl,
4609   print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4610   print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4611                     { \__enumext_filter_save_key:n {#1} }
4612                     \l__enumext_print_keyans_iv_tl,
4613   print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4614   print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4615                     { \__enumext_filter_save_key:n {#1} }
4616                     \l__enumext_print_keyans_vii_tl, % starred nested
4617   print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4618 }

```

- 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` 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`.

```

4619 \NewDocumentCommand \printkeyans { s O{} m }
4620 {
4621   \group_begin:
4622     \tl_use:N \l__enumext_print_keyans_i_tl
4623     \tl_use:N \l__enumext_print_keyans_ii_tl
4624     \tl_use:N \l__enumext_print_keyans_iii_tl

```

```

4625     \tl_use:N \l__enumext_print_keyans_iv_tl
4626     \tl_use:N \l__enumext_print_keyans_vii_tl
4627     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4628   \group_end:
4629 }

```

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

`\__enumext_printkeyans:nnn`

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.

```

4630 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4631 {
4632   \seq_if_exist:cTF { g__enumext_#3_seq }
4633   {
4634     \seq_if_empty:cF { g__enumext_#3_seq }
4635     {
4636       %%\seq_show:c { g__enumext_#3_seq }

```

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*`, it will open the environment `enumext*` passing the optional argument to the “first level”, set the key `base-fix` and then will map the `<sequence>`.

```

4637       \bool_if:nTF {#1}
4638       {
4639         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4640         {
4641           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4642         }
4643         {
4644           \tl_use:N \l__enumext_print_keyans_starred_tl
4645           \begin{enumext*}[#2]
4646             \keys_set:nn { enumext / level-1 }{ base-fix }
4647             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4648             \end{enumext*}
4649           }
4650       }

```

Otherwise it will open the environment `enumext` passing the optional argument to the “first level”, set the key `base-fix` and then map the `<sequence>`.

```

4651       {
4652         \begin{enumext}[#2]
4653         \keys_set:nn { enumext / enumext* }{ base-fix }
4654         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4655         \end{enumext}
4656       }
4657     }
4658   }
4659   {
4660     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4661   }
4662 }

```

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

## 12.47 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.

`\__enumext_filter_first_level:n`  
`\__enumext_filter_first_level_key:n`  
`\__enumext_filter_first_level_pair:nn`

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`.

```

4663 \cs_new:Npn \__enumext_filter_first_level:n #1
4664 {
4665   \use:e
4666   {
4667     \keyval_parse:NNn
4668     \__enumext_filter_first_level_key:n
4669     \__enumext_filter_first_level_pair:nn {#1}
4670   }
4671 }

```



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*`.

```

4672 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4673 {
4674   \str_case:nnF {#1}
4675   {
4676     { resume } {}
4677     { resume* } {}
4678   }
4679   { , { \exp_not:n {#1} } }
4680 }

```

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.

```

4681 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4682 {
4683   \str_case:nnF {#1}
4684   {
4685     { series } {}
4686     { resume } {}
4687     { save-ans } {}
4688   }
4689   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
4690 }

```

(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`.

```

4691 \keys_define:nn { enumext / meta-families }
4692 {
4693   enumext-1 .code:n =
4694     {
4695       \keys_set:ne { enumext / level-1 }
4696       {
4697         \__enumext_filter_first_level:n {#1}
4698       }
4699     } ,
4700   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4701   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4702   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4703   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4704   enumext* .code:n =
4705     {
4706       \keys_set:ne { enumext / enumext* }
4707       {
4708         \__enumext_filter_first_level:n {#1}
4709       }
4710     } ,
4711   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4712   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4713   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4714   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4715   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4716   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4717   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4718   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4719 }

```

We store them in the constant sequence `\c__enumext_all_families_seq` separated by commas.

```

4720 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4721 {
4722   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4723   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4724 }

```

`\setenumext` Now we define the user command `\setenumext`.

```

4725 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4726 {
4727   \seq_clear:N \l__enumext_setkey_tmpa_seq
4728   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4729   \int_set:Nn \l__enumext_setkey_tmpa_int

```

```

4730     {
4731       \seq_count:N \l__enumext_setkey_tmpb_seq
4732     }
4733     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4734     {
4735       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4736       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4737       \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4738       {
4739         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4740       }
4741     }
4742     {
4743       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4744     }
4745     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4746     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4747     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4748     {
4749       \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4750     }
4751   }

```

(End of definition for `\setenumext`. This function is documented on page 6.)

```

\__enumext_set_parse:n
\__enumext_set_error:nn

```

Internal functions used by the `\setenumext` command.

```

4752 \cs_new_protected:Npn \__enumext_set_parse:n #1
4753 {
4754   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4755   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4756   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4757   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4758   {
4759     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4760     { \tl_trim_spaces:n {#1} }
4761   }
4762   { \__enumext_set_error:nn {#1} { } }
4763 }
4764 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4765 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\__enumext_set_parse:n` and `\__enumext_set_error:nn`.)

## 12.48 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
```

First we will create a prop list `\c__enumext_meta_paths_prop` to handle the optional argument.

```

\c__enumext_meta_paths_prop
\__enumext_add_meta_key:nnn
\__enumext_def_meta_key:nnn
\__enumext_def_meta_key:Vnn

```

```

4766 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
4767 {
4768   {enumext,1} = level-1,
4769   {enumext,2} = level-2,
4770   {enumext,3} = level-3,
4771   {enumext,4} = level-4,
4772   {enumext*} = enumext*
4773 }

```

Now we create the user command taking care that unknown cannot be passed as an argument.

```

4774 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
4775 {
4776   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
4777   { \msg_error:nn { enumext } { prohibited-unknown } }
4778   {
4779     \bool_if:nTF {#1}
4780     {
4781       \int_step_inline:nn { 4 }
4782       { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
4783       \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
4784     }
4785     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }

```

```

4786     }
4787 }

```

The internal functions `\__enumext_add_meta_key:nnn` and `\__enumext_def_meta_key:nnn` will check the optional argument and create the “*meta-key*”.

```

4788 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
4789 {
4790   \tl_set:Nn \l__enumext_meta_path_tl {#1}
4791   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
4792   \prop_get:NVNTF
4793     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
4794     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
4795     {
4796       \msg_error:nnn { enumext } { unknown-set } {#1}
4797       \use_none:nn
4798     }
4799 }
4800 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
4801 {
4802   \bool_lazy_or:nnTF
4803     { \keys_if_exist:p:nn { enumext / #1 } {#2} }
4804     { \keys_if_exist:p:nn { enumext / enumext* } {#2} }
4805     { \msg_error:nnn { enumext } { already-defined } {#2} }
4806     {
4807       \keys_define:nn { enumext / #1 }
4808       {
4809         #2 .meta:n = {#3},
4810         #2 .value_forbidden:n = true
4811       }
4812     }
4813 }
4814 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

(End of definition for `\setenumextmeta` and others. This function is documented on page 6.)

## 12.49 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.

```

4815 \keys_define:nn { enumext / foreach }
4816 {
4817   before .tl_set:N = \l__enumext_foreach_before_tl,
4818   before .value_required:n = true,
4819   after .tl_set:N = \l__enumext_foreach_after_tl,
4820   after .value_required:n = true,
4821   start .int_set:N = \l__enumext_foreach_start_int,
4822   start .value_required:n = true,
4823   stop .int_set:N = \l__enumext_foreach_stop_int,
4824   stop .value_required:n = true,
4825   step .int_set:N = \l__enumext_foreach_step_int,
4826   step .value_required:n = true,
4827   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
4828   wrapper .value_required:n = true,
4829   sep .tl_set:N = \l__enumext_foreach_sep_tl,
4830   sep .value_required:n = true,
4831   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
4832 }
4833 \keys_precompile:nnN { enumext / foreach }
4834 {
4835   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
4836 }
4837 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown `<keys>`.

```

4838 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
4839 {
4840   \tl_if_blank:nTF {#2}
4841   {
4842     \msg_error:nnn { enumext } { for-key-unknown } {#1}

```

```

4843     }
4844     {
4845         \msg_error:nnn { enumext } { for-key-value-unknown } {#1} {#2}
4846     }
4847 }
4848 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
4849 {
4850     \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
4851 }

```

We create the command.

```

4852 \NewDocumentCommand \foreachkeyans { +0{ } m }
4853 {
4854     \__enumext_foreach_keyans:nn {#1} {#2}
4855 }

```

Finally the internal functions `\__enumext_foreach_keyans:nn` and `\__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

4856 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
4857 {
4858     \tl_use:N \g__enumext_foreach_default_keys_tl
4859     \keys_set:nn { enumext / foreach } {#1}
4860     \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
4861     \prop_if_exist:cF { g__enumext_#2_prop }
4862     {
4863         \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
4864     }
4865     \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
4866     {
4867         \int_set:Nn \l__enumext_foreach_stop_int
4868         { \prop_count:c { g__enumext_#2_prop } }
4869     }
4870     \seq_clear:N \l__enumext_foreach_print_seq
4871     \int_step_function:nnnN
4872     { \l__enumext_foreach_start_int }
4873     { \l__enumext_foreach_step_int }
4874     { \l__enumext_foreach_stop_int }
4875     \__enumext_foreach_add_body:n
4876     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
4877 }
4878 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
4879 {
4880     \seq_put_right:Ne \l__enumext_foreach_print_seq
4881     {
4882         \exp_not:V \l__enumext_foreach_before_tl
4883         \__enumext_foreach_wrapper:n
4884         {
4885             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
4886         }
4887         \exp_not:V \l__enumext_foreach_after_tl
4888     }
4889 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 16.)

## 12.50 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

4890 \msg_new:nnn { enumext } { package-load }
4891 {
4892     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4893 }
4894 \msg_new:nnn { enumext } { package-not-load }
4895 {
4896     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4897 }
4898 \msg_new:nnn { enumext } { package-load-foot }
4899 {
4900     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4901 }

```

Message used in the creation of counters by `enumext` package.

```

4902 \msg_new:nnn { enumext } { counters }

```

```

4903 {
4904     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4905     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4906 }

```

Message used by `align` and `mark-pos` keys.

```

4907 \msg_new:nnn { enumext } { unknown-choice }
4908 {
4909     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4910 }

```

Message used by reserved `anskey*` environment by `enumext` package.

```

4911 \msg_new:nnnn { enumext } { anskey-env-error }
4912 {
4913     The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4914     'enumext' ~ package, ~ It~ is~ already~ defined.
4915 }
4916 {
4917     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4918     for ~ the ~ 'save-ans' ~ key.\\
4919 }

```

Message used in the creation of `(prop list)` by `enumext` package.

```

4920 \msg_new:nnn { enumext } { store-prop }
4921 {
4922     * ~ Package ~ enumext: ~ Creating ~
4923     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4924 }
4925 \msg_new:nnn { enumext } { store-seq }
4926 {
4927     * ~ Package ~ enumext: ~ Creating ~
4928     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4929 }
4930 \msg_new:nnn { enumext } { store-int }
4931 {
4932     * ~ Package ~ enumext: ~ Creating ~
4933     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4934 }
4935 \msg_new:nnn { enumext } { prop-seq-int-hook }
4936 {
4937     * ~ Package ~ enumext: ~ Elements ~ in ~
4938     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4939     * ~ Package ~ enumext: ~ Elements ~ in ~
4940     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4941     * ~ Package ~ enumext: ~ Value ~ off ~
4942     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4943 }
4944 \msg_new:nnn { enumext } { item-answer-hook }
4945 {
4946     * ~ Package ~ enumext: ~ Value ~ off ~
4947     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4948     * ~ Package ~ enumext: ~ Value ~ off ~
4949     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4950     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4951 }

```

Message used by `[key = val]` system and `\setenumext` command.

```

4952 \msg_new:nnn { enumext } { invalid-key }
4953 {
4954     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4955 }
4956 \msg_new:nnn { enumext } { unknown-key-family }
4957 {
4958     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4959 }

```

Messages used in length calculation.

```

4960 \msg_new:nnn { enumext } { width-negative }
4961 {
4962     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4963     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ #2.
4964 }
4965 \msg_new:nnn { enumext } { width-zero }

```

```

4966 {
4967     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4968     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4969 }

```

Messages used by `show-length` key in `enumext`.

```

4970 \msg_new:nnn { enumext } { list-lengths }
4971 {
4972     **** ~ Lengths ~ used ~ by ~ '#2' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4973     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4974     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4975     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4976     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4977     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4978     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4979     \__enumext_show_length:nnn { skip } { topsep } {#1}
4980     \__enumext_show_length:nnn { skip } { parsep } {#1}
4981     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4982     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4983     *****
4984 }

```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```

4985 \msg_new:nnn { enumext } { list-lengths-not-nested }
4986 {
4987     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4988     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4989     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4990     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4991     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4992     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4993     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4994     \__enumext_show_length:nnn { skip } { topsep } {#1}
4995     \__enumext_show_length:nnn { skip } { parsep } {#1}
4996     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4997     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4998     *****
4999 }

```

Messages used by `ref` key.

```

5000 \msg_new:nnn { enumext } { key-ref-empty }
5001 {
5002     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5003 }

```

Messages used by `save-ans` key.

```

5004 \msg_new:nnn { enumext } { save-ans-empty }
5005 {
5006     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5007 }
5008 \msg_new:nnn { enumext } { save-ans-log }
5009 {
5010     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5011 }
5012 \msg_new:nnn { enumext } { save-ans-log-hook }
5013 {
5014     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5015 }
5016 \msg_new:nnn { enumext } { save-ans-hook }
5017 {
5018     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5019 }

```

Messages used by the internal system to check answer used by `check-ans` key.

```

5020 \msg_new:nnn { enumext } { need-save-ans }
5021 {
5022     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5023 }
5024 \msg_new:nnn { enumext } { items-same-answer }
5025 {
5026     *****\\
5027     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5028     for ~ \c_left_brace_str #2 \c_right_brace_str\\

```

```

5029     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5030     'OK', ~ all ~ items ~ with ~ answer.\\
5031     *****
5032 }
5033 \msg_new:nnn { enumext } { item-greater-answer }
5034 {
5035     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5036     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5037     Items ~ > ~ Answers.
5038 }
5039 \msg_new:nnn { enumext } { item-less-answer }
5040 {
5041     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5042     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5043     Items ~ < ~ Answers.
5044 }

```

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```

5045 \msg_new:nnn { enumext } { missing-starred }
5046 {
5047     Missing ~ '\c_backslash_str #1*' ~ #2.
5048 }
5049 \msg_new:nnn { enumext } { many-starred }
5050 {
5051     Many ~ '\c_backslash_str #1*' ~ #2.
5052 }

```

Messages used by `\printkeyans*` command.

```

5053 \msg_new:nnn { enumext } { print-starred }
5054 {
5055     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5056     #2 ~ environment ~ \msg_line_context:.
5057 }

```

Message for the nesting depth of the environment `enumext`.

```

5058 \msg_new:nnn { enumext } { list-too-deep }
5059 {
5060     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
5061     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5062 }

```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```

5063 \msg_new:nnn { enumext } { anskey-unnumber-item }
5064 {
5065     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5066 }
5067 \msg_new:nnn { enumext } { anskey-already-stored }
5068 {
5069     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5070 }
5071 \msg_new:nnn { enumext } { anskey-empty-arg }
5072 {
5073     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5074 }
5075 \msg_new:nnn { enumext } { anskey-wrong-place }
5076 {
5077     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
5078     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5079 }
5080 \msg_new:nnn { enumext } { anskey-nested }
5081 {
5082     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5083 }
5084 \msg_new:nnn { enumext } { anskey-math-mode }
5085 {
5086     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5087 }
5088 \msg_new:nnn { enumext } { anskey-env-wrong }
5089 {
5090     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5091 }
5092 \msg_new:nnn { enumext } { anspic-wrong-place }
5093 {

```



```

5094     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5095     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5096 }
5097 \msg_new:nnn { enumext } { command-wrong-place }
5098 {
5099     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5100     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5101 }
5102 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5103 {
5104     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5105     'anskey*' ~ and ~ is ~ being ~ ignored.
5106 }
5107 {
5108     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5109     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5110 }
5111 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5112 {
5113     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5114     'anskey*' ~ and ~ is ~ being ~ ignored.
5115 }
5116 {
5117     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5118     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5119 }
5120 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5121 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5122 {
5123     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5124     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5125 }
5126 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5127 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5128 {
5129     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5130     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5131 }

```

Messages used by `keyans`, `keyans*` and `keyanspic` environment.

```

5132 \msg_new:nnn { enumext } { keyans-nested }
5133 {
5134     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5135 }
5136 \msg_new:nnn { enumext } { keyans-wrong-level }
5137 {
5138     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5139     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5140 }
5141 \msg_new:nnn { enumext } { wrong-place }
5142 {
5143     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5144     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5145 }
5146 \msg_new:nnn { enumext } { keyanspic-nested }
5147 {
5148     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
5149 }
5150 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5151 {
5152     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5153     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5154 }
5155 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5156 {
5157     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5158 }
5159 \msg_new:nnnn { enumext } { keyans-unknown-key }
5160 {
5161     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5162     '\_enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5163 }

```

```

5164 {
5165     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5166     ~ have ~ a ~ key ~ called ~ '#1'.\\
5167     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5168 }
5169 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5170 {
5171     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5172     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5173 }
5174 {
5175     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5176     ~ have ~ a ~ key ~ called ~ '#1'.\\
5177     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5178 }

```

Message used by unknown  $\langle keys \rangle$  in `enumext*`. environment.

```

5179 \msg_new:nnnn { enumext } { starred-unknown-key }
5180 {
5181     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5182     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5183 }
5184 {
5185     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5186     ~ have ~ a ~ key ~ called ~ '#1'.\\
5187     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5188 }
5189 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5190 {
5191     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5192     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5193 }
5194 {
5195     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5196     ~ have ~ a ~ key ~ called ~ '#1'.\\
5197     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5198 }

```

Message used by unknown  $\langle keys \rangle$  in `enumext` environment.

```

5199 \msg_new:nnnn { enumext } { standar-unknown-key }
5200 {
5201     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5202     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5203 }
5204 {
5205     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5206     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5207     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5208 }
5209 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5210 {
5211     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5212     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5213 }
5214 {
5215     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5216     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5217     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5218 }

```

Message used by unknown  $\langle keys \rangle$  in `\foreachkeyans`.

```

5219 \msg_new:nnnn { enumext } { for-key-unknown }
5220 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5221 {
5222     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5223     Check~that~you~have~spelled~the~key~name~correctly.
5224 }
5225 \msg_new:nnnn { enumext } { for-key-value-unknown }
5226 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5227 {
5228     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5229     Check~that~you~have~spelled~the~key~name~correctly.
5230 }

```

Messages used by `\getkeyans` command.

```
5231 \msg_new:nnn { enumext } { undefined-storage-anskey }
5232 {
5233   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5234 }
```

Messages used by `\miniright` command.

```
5235 \msg_new:nnn { enumext } { missing-miniright }
5236 {
5237   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
5238   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5239 }
5240 \msg_new:nnn { enumext } { wrong-miniright-place }
5241 {
5242   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.\
5243   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5244 }
5245 \msg_new:nnn { enumext } { wrong-miniright-use }
5246 {
5247   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.\
5248   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5249 }
5250 \msg_new:nnn { enumext } { wrong-miniright-starred }
5251 {
5252   Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5253 }
5254 \msg_new:nnn { enumext } { many-miniright-used }
5255 {
5256   Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5257 }
```

Messages used by `\setenumextmeta` command.

```
5258 \msg_new:nnn { enumext } { unknown-set }
5259 {
5260   Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5261 }
5262 \msg_new:nnn { enumext } { already-defined }
5263 {
5264   The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5265 }
5266 \msg_new:nnn { enumext } { prohibited-unknown }
5267 {
5268   The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5269 }
```

Messages used by `enumext*` and `keyans*` environments.

```
5270 \msg_new:nnn { enumext } { nested }
5271 {
5272   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_context:.
5273 }
5274 \msg_new:nnn { enumext } { nested-horizontal }
5275 {
5276   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
5277 }
5278 \msg_new:nnn { enumext } { item-joined }
5279 {
5280   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5281 }
5282 \msg_new:nnn { enumext } { item-joined-columns }
5283 {
5284   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5285 }
```

## 12.51 Finish package

Finish package implementation.

```
5286 \file_input_stop:
5287 </package>
```

### 13 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> .....	219
<code>\+</code> .....	211
<code>\-</code> .....	211
<code>\\</code> 227, 2666, 3667, 4904, 4913, 4918, 4938, 4940, 4947, 4949, 4962, 4967, 4972, 4987, 5026, 5028, 5030, 5035, 5036, 5041, 5042, 5060, 5077, 5094, 5099, 5108, 5117, 5123, 5129, 5138, 5143, 5152, 5166, 5176, 5186, 5196, 5206, 5216, 5222, 5228, 5237, 5242, 5247	
A	
above .....	<u>1481</u>
above* .....	<u>1481</u>
<code>\addvspace</code> 1148, 1177, 1277, 1325, 1388, 1394, 1431, 1439, 1467, 3480, 3484, 3620, 3636, 3994, 4008, 4049, 4063	
after .....	<u>960</u>
align .....	<u>509</u>
<code>\Alph</code> .....	36, <u>41</u>
<code>\Alph</code> .....	461, 576, 621, 689, 4613
<code>\alph</code> .....	36, <u>41</u>
<code>\alph</code> .....	462, 574, 4605
<code>\anskey</code> .....	12, 73, 75, <u>2484</u>
<code>anskey*</code> .....	13, <u>2594</u>
<code>\anspic</code> .....	15, 99, <u>3645</u>
<code>\anspic*</code> .....	67
<code>\arabic</code> .....	30, <u>36</u>
<code>\arabic</code> .....	460, 573, 620, 4597, 4601, 4617
B	
base-fix .....	<u>819</u>
<code>\baselineskip</code> .....	<u>50</u>
<code>\baselineskip</code> .....	836, <u>847</u>
before .....	<u>960</u>
before* .....	<u>960</u>
below .....	<u>1481</u>
below* .....	<u>1481</u>
bool commands:	
<code>\bool_gset_false:N</code> 332, 333, 334, 2770, 2772, 4010, 4014, 4065	
<code>\bool_gset_true:N</code> 240, 250, 1063, 1974, 1980, 3986, 4011, 4041, 4066	
<code>\bool_if:NTF</code> . 400, 412, 429, 1410, 1503, 1517, 1530, 1541, 1552, 1563, 1574, 1585, 1634, 1651, 1656, 1664, 1691, 1729, 1734, 1741, 1745, 1767, 1772, 1780, 1787, 1818, 1826, 1919, 2117, 2127, 2206, 2230, 2237, 2261, 2359, 2381, 2421, 2434, 2438, 2488, 2507, 2531, 2585, 2596, 2685, 2722, 2786, 2819, 2834, 2909, 2920, 2924, 2943, 2956, 2998, 3032, 3067, 3201, 3263, 3273, 3305, 3310, 3413, 3461, 3476, 3490, 3549, 3604, 3618, 3626, 3647, 3982, 3991, 3995, 4037, 4046, 4050, 4139, 4149, 4232, 4237, 4246, 4250, 4265, 4294, 4350, 4452, 4456, 4480, 4489, 4493, 4499, 4513, 4536	
<code>\bool_if:nTF</code> 1440, 1468, 3054, 3183, 3221, 3668, 4637, 4779	
<code>\bool_if_p:N</code> 259, 274, 832, 833, 843, 844, 1798, 1799, 1807, 1808, 1932, 1958, 1971, 1972, 1977, 1978, 2294, 2304, 2316, 2331, 2332, 2366, 2407, 2408, 2708, 2896, 2897, 2934, 2935, 3386, 3388, 3399, 3675, 3676	
<code>\bool_lazy_all:nTF</code> 257, 272, 1930, 1956, 2292, 2301, 2314, 2329, 3384, 3397	
<code>\bool_lazy_and:nnTF</code> 236, 246, 831, 842, 1403, 1797, 1806, 1970, 1976, 2365, 2372, 2406, 2549, 2561, 2707, 2713, 2895	
<code>\bool_lazy_or:nnTF</code> . . 1859, 1866, 2933, 3674, 4802	
<code>\bool_new:N</code> 34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 95, 100, 101, 106, 107, 110, 135, 136, 144, 145, 150, 152, 153, 167, 179, 181	
<code>\bool_not_p:n</code> 237, 247, 2303, 2367, 2373, 2709, 2714, 3387, 3400	
<code>\bool_set_eq:NN</code> . . . . . 3007, 3163, 4182, 4406	
<code>\bool_set_false:N</code> 409, 853, 1904, 1905, 1937, 1942, 1946, 1950, 1963, 2649, 3361, 3507, 3557, 3641, 3706, 3724, 4107, 4134, 4179, 4356, 4403	
<code>\bool_set_true:N</code> . 264, 265, 279, 280, 391, 395, 502, 868, 1487, 1492, 1754, 1876, 1877, 2149, 2157, 2650, 3001, 3003, 3035, 3037, 3159, 3171, 3285, 3360, 3393, 3406, 3432, 3554, 3581, 3971, 4026, 4106, 4188, 4195, 4196, 4240, 4354, 4412, 4419, 4420	
box commands:	
<code>\box_dp:N</code> . . 1334, 1335, 1338, 1345, 1358, 1366, 1372, 1380, 3736	
<code>\box_ht:N</code> . . 1205, 1208, 1219, 1220, 1228, 1229, 1240, 1241, 1248, 1257, 1301, 1325	
<code>\box_new:N</code> . . . . . 70, 174, 180	
<code>\box_set_wd:Nn</code> . . . . . 4286, 4528	
<code>\box_use_drop:N</code> . . . . . 4006, 4061, 4293, 4535	
<code>\box_wd:N</code> . . . . . 468	
C	
<code>\c</code> .....	219, 220, 726, 728, 740, 742
<code>\catcode</code> .....	2666
<code>\cB</code> .....	220
<code>\cE</code> .....	220
<code>\centering</code> .....	1442, 1470, 3762, 3999, 4054
check-ans .....	<u>1896</u>
Document class:	
article .....	42
clist commands:	
<code>\clist_const:Nn</code> .....	186
<code>\clist_map_function:nN</code> .....	3749
<code>\clist_map_inline:Nn</code> 508, 774, 959, 974, 1055, 1497	
<code>\clist_map_inline:nn</code> . 49, 60, 78, 85, 97, 109, 138, 161, 185, 536, 556, 828, 873, 894, 1069, 1603, 1843, 1910, 2096, 2114, 2146, 2289, 2828, 3088, 3100, 3140, 3250, 3253, 3280, 3292, 3295, 3315, 4755	
<code>\columnbreak</code> .....	73
<code>\columnbreak</code> .....	2369
columns .....	<u>1039</u>
columns-sep .....	<u>1039</u>
<code>\columnsep</code> .....	95
<code>\columnsep</code> .....	3456, 3602
<code>\columnseprule</code> .....	95
<code>\columnseprule</code> .....	3459, 3603
Commands provide by <b>enumext</b> :	
<code>\anskey</code> 28, 63, 64, 69, 70, 72–76, 82, 84, 94, 109, 117, 118, 126	
<code>\anspic*</code> . . 28, 29, 67, 70, 82, 83, 99, 100, 102, 117, 118	
<code>\anspic</code> .....	70, 99–101, 126
<code>\foreachkeyans</code> .....	122, 128
<code>\getkeyans</code> .....	70, 117, 129

<code>\item*</code>	28, 29, 67, 70, 82, 83, 85, 86, 89, 110, 115, 117, 118
<code>\item</code>	85, 89, 104, 109, 110, 114
<code>\miniright</code>	27, 47, 54, 55, 95, 96, 129
<code>\printkeyans*</code>	118
<code>\printkeyans</code>	28, 70, 118
<code>\setenumextmeta</code>	121, 129
<code>\setenumext</code>	28, 118, 120, 121, 124
Counters defined by <code>enumext</code> :	
<code>enumXiii</code>	26, 36
<code>enumXii</code>	26, 36
<code>enumXiv</code>	26, 36
<code>enumXi</code>	26, 36
<code>enumXviii</code>	26, 36
<code>enumXvii</code>	26, 36, 110
<code>enumXvi</code>	26, 36
<code>enumXv</code>	26, 36
cs commands:	
<code>\cs_generate_variant:Nn</code>	191, 192, 470, 486, 732, 748, 2198, 2203, 2279, 2602, 3240, 3751, 4814
<code>\cs_if_exist:NTF</code>	440
<code>\cs_if_free:NTF</code>	2553, 2565
<code>\cs_new:Nn</code>	205
<code>\cs_new:Npn</code>	223, 1604, 1613, 1621, 2161, 2170, 2178, 4663, 4672, 4681
<code>\cs_new_eq:NN</code>	359, 360, 361, 365, 366, 414, 415, 418, 419
<code>\cs_new_protected:Nn</code>	215, 229, 255, 288, 318, 324, 330, 336, 342, 350, 368, 386, 597, 660, 712, 829, 975, 979, 983, 987, 991, 995, 999, 1003, 1007, 1011, 1015, 1019, 1023, 1027, 1031, 1035, 1096, 1108, 1132, 1150, 1161, 1179, 1211, 1265, 1279, 1314, 1327, 1349, 1384, 1390, 1498, 1512, 1526, 1537, 1548, 1559, 1570, 1581, 1662, 1765, 1778, 1795, 1816, 1844, 1849, 1874, 1915, 1925, 1968, 1983, 1990, 1999, 2004, 2009, 2014, 2023, 2028, 2033, 2204, 2228, 2235, 2259, 2266, 2280, 2505, 2524, 2540, 2603, 2639, 2670, 2705, 2747, 2768, 2776, 2817, 2832, 2860, 2893, 2929, 2941, 2954, 3040, 3050, 3061, 3179, 3195, 3336, 3353, 3382, 3411, 3418, 3440, 3470, 3488, 3530, 3547, 3571, 3588, 3613, 3624, 3664, 3708, 3722, 3747, 3752, 3768, 3772, 3791, 3801, 3832, 3961, 3980, 4016, 4035, 4093, 4121, 4128, 4137, 4147, 4164, 4305, 4342, 4369, 4375, 4388, 4444, 4548
<code>\cs_new_protected:Npn</code>	193, 197, 201, 422, 438, 455, 465, 471, 577, 622, 694, 719, 733, 1070, 1420, 1453, 1630, 1649, 1719, 1752, 1854, 2038, 2115, 2125, 2147, 2155, 2190, 2199, 2355, 2418, 2432, 2470, 2474, 2594, 2625, 2629, 2660, 2796, 2870, 2914, 2994, 3013, 3101, 3105, 3119, 3123, 3141, 3145, 3155, 3167, 3209, 3243, 3283, 3364, 3567, 3717, 3863, 3912, 4110, 4170, 4177, 4193, 4201, 4206, 4218, 4362, 4394, 4401, 4417, 4425, 4439, 4569, 4582, 4630, 4752, 4764, 4788, 4800, 4838, 4848, 4856, 4878
<code>\cs_new_protected_nopar:Nn</code>	4157, 4281, 4381, 4523
<code>\cs_new_protected_nopar:Npn</code>	4224, 4472
<code>\cs_set:Npn</code>	2290, 2327, 4575
<code>\cs_set_eq:NN</code>	4083, 4084, 4226, 4332, 4333, 4474
<code>\cs_set_protected:Nn</code>	899, 915, 927, 939
<code>\cs_set_protected:Npn</code>	45, 54, 71, 79, 92, 98, 131, 157, 165, 487, 509, 541, 557, 604, 749, 775, 819, 855, 878, 951, 960, 1039, 1056, 1481, 1592, 1835, 1896, 2055, 2097, 2133, 2282, 2821, 3077, 3093, 3133, 3241, 3281
<code>\cs_to_str:N</code>	457, 480
<code>\cs_undefine:N</code>	2542, 2543, 2544, 2545
<b>D</b>	
<code>\d</code>	211
<code>\DeclareDocumentEnvironment</code>	372
dim commands:	
<code>\dim_abs:n</code>	3214, 3219
<code>\dim_add:Nn</code>	3727, 3826, 3857
<code>\dim_compare:nNnTF</code>	901, 917, 929, 941, 1223, 1231, 1422, 1455, 3211, 3216, 3222, 3228, 3230, 3232, 3423, 3445, 3575, 3592, 3719, 3803, 3819, 3834, 3850, 3963, 4018
<code>\dim_compare:nTF</code>	2391, 2735, 3342, 3536
<code>\dim_gset_eq:NN</code>	3972, 4027
<code>\dim_gzero:N</code>	2774, 4013, 4068
<code>\dim_new:N</code>	67, 74, 75, 76, 94, 140, 173, 175, 176, 182
<code>\dim_set:Nn</code>	468, 869, 3030, 3214, 3219, 3221, 3224, 3225, 3229, 3231, 3234, 3235, 3237, 3338, 3426, 3448, 3532, 3577, 3594, 3754, 3805, 3812, 3836, 3843, 3898, 3947, 3965, 4020, 4220
<code>\dim_set_eq:NN</code>	564, 611, 682, 686, 2945, 2946, 2958, 2959, 3025, 3252, 3294, 3456, 3602, 3905, 3908, 3909, 3954, 3957, 3958, 4211
<code>\dim_sub:Nn</code>	3347, 3541, 3821, 3852
<code>\dim_use:N</code>	902, 910, 1423, 1436, 2269, 2272, 2277, 3045, 3047, 3344, 3349, 3424, 3429, 3430, 3436, 3446, 3450, 3451, 3453
<code>\dim_zero:N</code>	3286, 3459, 3603, 3728, 3729, 3730
<code>\dim_zero_new:N</code>	437
<code>\c_zero_dim</code>	904, 918, 930, 942, 1423, 1455, 2393, 2737, 3211, 3216, 3222, 3229, 3344, 3424, 3446, 3538, 3575, 3592, 3803, 3819, 3834, 3850, 3963, 4018
<code>\dimeval</code>	2062
<b>E</b>	
<code>\end</code>	1433, 1462, 2232, 2263, 3475, 3499, 3617, 3635, 3984, 4007, 4039, 4062, 4639, 4648, 4655
<code>\endgroup</code>	2666
<code>\endlist</code>	360
<code>\endlrbox</code>	4284, 4526
<code>\endminipage</code>	366
<code>enumext</code>	5, <u>3316</u>
enumext internal commands:	
<code>\l__enumext_ref_the_count_tl</code>	38
<code>\l__enumext_resume_name_tl</code>	59
<code>\__enumext_add_meta_key:nnn</code>	122, <u>4766</u> , 4782, 4783, 4785, 4788
<code>\__enumext_add_pre_parsep:</code>	48, 1106, <u>1108</u> , 1108
<code>\__enumext_after_args_exec:</code>	46, <u>975</u> , 987, 3329
<code>\__enumext_after_args_exec_v:</code>	<u>991</u> , 1003, 3523
<code>\__enumext_after_args_exec_vii:</code>	<u>1007</u> , 1031
<code>\__enumext_after_args_exec_viii:</code>	1035
<code>\__enumext_after_env:nn</code>	79, 80, 82, 96, 105, 112, 197, 197, 2680, 3510, 3989, 4044, 4319
<code>\__enumext_after_hyperref:</code>	34, 384, <u>386</u> , 386
<code>\__enumext_after_list:</code>	96, 113, 3334, <u>3488</u> , 3488
<code>\l__enumext_after_list_args_v_tl</code>	1005
<code>\l__enumext_after_list_args_vii_tl</code>	1033, 4275
<code>\l__enumext_after_list_args_viii_tl</code>	1037, 4509
<code>\__enumext_after_list_v:</code>	3528, <u>3571</u> , 3624
<code>\__enumext_after_list_vii:</code>	108, 4091, <u>4128</u> , 4128
<code>\__enumext_after_list_viii:</code>	4340, <u>4375</u> , 4375
<code>\__enumext_after_stop_list:</code>	46, 96, <u>975</u> , 983, 3504
<code>\__enumext_after_stop_list_v:</code>	<u>991</u> , 999, 3642
<code>\l__enumext_after_stop_list_v_tl</code>	1001

```

\__enumext_after_stop_list_vii: .. 108, 1007,
    1023, 4131
\l__enumext_after_stop_list_vii_tl ... 1025
\__enumext_after_stop_list_viii: . 1027, 4378
\l__enumext_after_stop_list_viii_tl ... 1029
\l__enumext_align_label_vii_str .. 4267, 4271
\l__enumext_align_label_viii_str . 4501, 4505
\l__enumext_align_label_X_str ..... 165
\c__enumext_all_envs_clist .. 186, 508, 774, 959,
    974, 1055, 1497
\c__enumext_all_families_seq .. 120, 4720, 4746
\l__enumext_anskey_env_bool 31, 78, 34, 265, 280,
    2596
\__enumext_anskey_env_clean_vars: . 81, 2701,
    2705, 2768
\__enumext_anskey_env_define_keys: 78, 2594,
    2603, 2674
\__enumext_anskey_env_exec: 79, 2599, 2670, 2670
\__enumext_anskey_env_make:n 63, 78, 1879, 2594,
    2594, 2602
\__enumext_anskey_env_reset_keys: 79, 80, 2639,
    2702
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2594
\__enumext_anskey_env_save_keys: .. 80, 2682,
    2705, 2705
\__enumext_anskey_env_store: .. 81, 2698, 2705,
    2747
\__enumext_anskey_env_unknown:n 79, 2622, 2625
\__enumext_anskey_env_unknown:nn . 2627, 2629
\l__enumext_anskey_level_int .. 28, 2526, 2527
\__enumext_anskey_safe_inner: . 76, 2499, 2505,
    2524
\__enumext_anskey_safe_inner:n ..... 76
\__enumext_anskey_safe_outer: . 76, 2486, 2505,
    2505
\__enumext_anskey_show_wrap_arg:n . 74, 2418,
    2418, 2436, 2451
\__enumext_anskey_show_wrap_left:n 75, 2363,
    2432, 2432
\__enumext_anskey_unknown:n 75, 2454, 2468, 2470
\__enumext_anskey_unknown:nn . 2454, 2472, 2474
\__enumext_anskey_wrapper:n ..... 2059, 2430
\__enumext_at_begin_document:n .. 33, 193, 193,
    357, 363
\l__enumext_base_line_fix_bool . 823, 833, 844,
    853
\__enumext_before_args_exec: .. 46, 94, 108, 975,
    975, 3421
\__enumext_before_args_exec_v: 991, 991, 3574
\__enumext_before_args_exec_vii: . 1007, 1007,
    4125
\__enumext_before_args_exec_viii: 1011, 4372
\__enumext_before_env:nn 78, 197, 201, 2547, 2559,
    2571, 2672
\__enumext_before_keys_exec: 46, 975, 979, 3326
\__enumext_before_keys_exec_v: 991, 995, 3520
\__enumext_before_keys_exec_vii ..... 1007
\__enumext_before_keys_exec_vii: . 1015, 4079
\__enumext_before_keys_exec_viii: 1019, 4328
\__enumext_before_list: ... 94, 3320, 3418, 3418
\__enumext_before_list_v: ... 3515, 3571, 3571
\__enumext_before_list_vii: ... 108, 4074, 4121,
    4121
\__enumext_before_list_viii: ... 113, 4324, 4369,
    4369
\l__enumext_before_no_starred_key_v_tl 997
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1017
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1021
\l__enumext_before_starred_key_v_tl ... 993
\l__enumext_before_starred_key_vii_tl . 1009
\l__enumext_before_starred_key_viii_tl 1013
\__enumext_calc_hspace:NNNNNNN 90, 3209, 3209,
    3240, 3245, 3287
\__enumext_check_ans_active: . 65, 94, 108, 1915,
    1915, 3422, 4124
\g__enumext_check_ans_item_tl ..... 84
\g__enumext_check_ans_key_bool 66, 67, 144, 332,
    1974, 1980, 2786
\l__enumext_check_ans_key_bool 66, 1900, 1905,
    1971, 1977
\__enumext_check_ans_key_hook: 66, 96, 108, 1968,
    1968, 3505, 4132
\__enumext_check_ans_level: 65, 1915, 1921, 1925
\__enumext_check_ans_log: 66, 67, 81, 2014, 2014,
    2790
\__enumext_check_ans_log_msg_greater: 2014,
    2020, 2033
\__enumext_check_ans_log_msg_less: 2014, 2018,
    2023
\__enumext_check_ans_log_msg_same_ok: 2014,
    2019, 2028
\__enumext_check_ans_msg_greater: 1990, 1996,
    2009
\__enumext_check_ans_msg_less: 1990, 1994, 1999
\__enumext_check_ans_msg_same_ok: 1990, 1995,
    2004
\__enumext_check_ans_show: .. 66, 81, 1990, 1990,
    2788
\l__enumext_check_answers_bool 63, 65, 76, 85, 86,
    144, 1877, 1904, 1919, 2206, 2230, 2237, 2261, 2488,
    2685, 2909, 2998, 3032, 4237
\__enumext_check_starred_cmd:n 32, 67, 84, 2038,
    2038, 3526, 3703, 4338
\g__enumext_check_starred_cmd_int 144, 2041,
    2047, 2052, 3177, 3673, 4451
\l__enumext_check_start_line_env_tl . 32, 144,
    295, 303, 311, 2044, 2050, 2053
\l__enumext_columns_sep_v_dim 3592, 3594, 3602
\l__enumext_columns_sep_vii_dim .. 3803, 3805,
    3814, 3826, 3902, 4303
\l__enumext_columns_sep_viii_dim . 3834, 3836,
    3845, 3857, 3951, 4546
\l__enumext_columns_v_int 1308, 1458, 3590, 3598,
    3610, 3615
\l__enumext_columns_vii_int .. 3808, 3811, 3815,
    3824, 3866, 3870, 3873, 3879, 3885, 3889, 4298, 4309
\l__enumext_columns_viii_int . 3839, 3842, 3846,
    3855, 3915, 3919, 3922, 3928, 3934, 3938, 4541, 4554
\l__enumext_counter_i_tl ..... 45, 447
\l__enumext_counter_ii_tl ..... 45, 448
\l__enumext_counter_iii_tl ..... 45, 449
\l__enumext_counter_iv_tl ..... 45, 450
\c__enumext_counter_style_tl ..... 30, 50, 217
\g__enumext_counter_styles_tl . 26, 36, 67, 458,
    476
\l__enumext_counter_v_tl ..... 45, 451, 702

```



`\l__enumext_counter_vii_tl` . . . . . 45, 452  
`\l__enumext_counter_viii_tl` . . . . . 45, 453, 632  
`\l__enumext_counter_viii_tl` . . . . . 45, 454, 649  
`\l__enumext_current_widest_dim` 26, 67, 482, 565, 612, 683, 687  
`\__enumext_def_meta_key:nnn` . . . 122, 4766, 4794, 4800, 4814  
`\__enumext_default_item:n` . . . 2994, 2994, 3058  
`\__enumext_define_counters:Nn` 26, 438, 438, 447, 448, 449, 450, 451, 452, 453, 454  
`\__enumext_endminipage:` . 33, 363, 366, 380, 3764, 4283, 4525  
`\g__enumext_envir_name_tl` 31, 34, 266, 281, 340, 1847, 1852, 1862, 2002, 2007, 2012, 2026, 2031, 2036  
`\l__enumext_envir_name_tl` . 31, 32, 34, 235, 245, 294, 302, 310, 5162, 5165, 5172, 5175, 5182, 5185, 5192, 5195, 5201, 5205, 5211, 5215, 5272, 5276  
`\__enumext_execute_after_env:` 32, 33, 63, 66, 67, 77, 81, 2776, 2776, 3510, 4319  
`\__enumext_fake_item:` . . . . . 899, 899, 3272  
`\l__enumext_fake_item_indent_v_dim` 918, 923  
`\l__enumext_fake_item_indent_v_tl` 920, 3160, 3164, 3172  
`\l__enumext_fake_item_indent_vii_dim` 930, 935  
`\l__enumext_fake_item_indent_vii_tl` 932, 4279  
`\l__enumext_fake_item_indent_viii_dim` . 942, 947, 4517  
`\l__enumext_fake_item_indent_viii_tl` . . 944, 4515, 4520  
`\l__enumext_fake_item_indent_X_tl` . . . . . 98  
`\__enumext_fake_item_vii:` . . . . 899, 927, 3304  
`\__enumext_fake_item_viii:` . . . . 899, 939, 3309  
`\__enumext_filter_first_level:n` . . 119, 4663, 4663, 4697, 4708  
`\__enumext_filter_first_level_key:n` 120, 4663, 4668, 4672  
`\__enumext_filter_first_level_pair:nn` . 120, 4663, 4669, 4681  
`\__enumext_filter_save_key:n` . . 69, 2122, 2130, 2153, 2159, 2161, 2161, 4595, 4599, 4603, 4607, 4611, 4615  
`\__enumext_filter_save_key_key:n` . . 70, 2161, 2166, 2170  
`\__enumext_filter_save_key_pair:nn` 70, 2161, 2167, 2178  
`\__enumext_filter_series:n` 58, 1604, 1604, 1642, 1654, 1659  
`\__enumext_filter_series_key:n` 58, 1604, 1609, 1613  
`\__enumext_filter_series_pair:nn` . . 58, 1604, 1610, 1621  
`\g__enumext_footnote_arg_seq` . 162, 3774, 3787, 3797  
`\g__enumext_footnote_int` . 162, 3781, 3784, 3786, 3788  
`\g__enumext_footnote_int_seq` . 162, 3775, 3788, 3793, 3796  
`\__enumext_footnotes_key_bool` . . . . . 34  
`\l__enumext_footnotes_key_bool` 29, 34, 111, 152, 395, 400, 409, 4246, 4294, 4489, 4536  
`\__enumext_footnotetext:nn` . . . 3768, 3768, 3798  
`\__enumext_foreach_add_body:n` . 123, 4815, 4875, 4878  
`\l__enumext_foreach_after_tl` . . . . . 4819, 4887  
`\l__enumext_foreach_before_tl` . . . . 4817, 4882  
`\g__enumext_foreach_default_keys_tl` 122, 124, 4837, 4858  
`\__enumext_foreach_keyans:nn` . . 123, 4815, 4854, 4856  
`\l__enumext_foreach_name_prop_tl` . 124, 4860, 4885  
`\l__enumext_foreach_print_seq` 124, 4870, 4876, 4880  
`\l__enumext_foreach_sep_tl` . . . . . 4829, 4876  
`\l__enumext_foreach_start_int` . . . . 4821, 4872  
`\l__enumext_foreach_step_int` . . . . 4825, 4873  
`\l__enumext_foreach_stop_int` . 4823, 4865, 4867, 4874  
`\__enumext_foreach_wrapper:n` . . . . . 4827, 4883  
`\__enumext_getkeyans:nn` . . 118, 4578, 4582, 4582  
`\__enumext_getkeyans_aux:n` 117, 4566, 4569, 4569  
`\l__enumext_hyperref_bool` . 29, 34, 35, 152, 391, 412, 429, 2408, 2897, 4232, 4480  
`\__enumext_hypertarget:nn` 35, 386, 414, 418, 434  
`\__enumext_if_is_int:n` . . . . . 209  
`\__enumext_if_is_int:nTF` . . . . . 209, 721, 735  
`\__enumext_internal_mini_page:` 34, 93, 107, 368, 368, 3355, 4095  
`\__enumext_is_not_nested:` 26, 31, 93, 107, 229, 229, 3356, 4096  
`\__enumext_is_on_first_level:` . 26, 31, 93, 107, 229, 255, 3362, 4108  
`\g__enumext_item_anskey_int` 76, 84, 144, 327, 354, 355, 1987, 2357, 2911  
`\__enumext_item_answer_diff:` . . 66, 67, 81, 1983, 1983, 2783  
`\g__enumext_item_answer_diff_int` . 66, 67, 144, 328, 1985, 1992, 2016  
`\l__enumext_item_column_pos_vii_int` 109, 3873, 3879, 3885, 3889, 3896, 4160, 4298, 4301  
`\l__enumext_item_column_pos_viii_int` . . 114, 3922, 3928, 3934, 3938, 3945, 4384, 4541, 4544  
`\l__enumext_item_column_pos_X_int` . . . . . 165  
`\g__enumext_item_count_all_vii_int` 109, 3897, 4161, 4309, 4316  
`\g__enumext_item_count_all_viii_int` 114, 3946, 4385, 4553, 4561  
`\g__enumext_item_count_all_X_int` . . . . . 165  
`\g__enumext_item_number_bool` . . . . . 144  
`\l__enumext_item_number_bool` 65, 150, 1937, 1942, 1946, 1950, 1963, 2531, 2585, 3001, 3035, 4240  
`\g__enumext_item_number_int` . . 65, 144, 326, 353, 355, 1936, 1941, 1945, 1949, 1962, 1987, 3000, 3034, 4239  
`\__enumext_item_peek_args_vii:` 109, 4162, 4164, 4164  
`\__enumext_item_peek_args_viii:` . . 114, 4386, 4388, 4388  
`\__enumext_item_star_exec:` . 86, 3013, 3040, 3069  
`\l__enumext_item_starred_vii_bool` 4179, 4195, 4250  
`\l__enumext_item_starred_viii_bool` 4403, 4419, 4493, 4513  
`\l__enumext_item_starred_X_bool` . . . . . 165  
`\__enumext_item_std:w` 33, 85, 86, 89, 101, 357, 361, 3004, 3010, 3038, 3160, 3164, 3172, 3740  
`\g__enumext_item_symbol_aux_tl` . 86, 128, 3018, 3021, 3046, 3074



```

\g__enumext_item_symbol_aux_vii_tl 4203, 4252,
    4255, 4259, 4261
\g__enumext_item_symbol_aux_X_tl . . . . . 165
\l__enumext_item_symbol_sep_vii_dim . . 4212,
    4220, 4258, 4260
\l__enumext_item_symbol_vii_tl . . . . . 4255
\l__enumext_item_text_vii_box 4245, 4286, 4293
\l__enumext_item_text_viii_box 4488, 4528, 4535
\l__enumext_item_text_X_box . . . . . 165
\l__enumext_item_width_vii_dim . . . 3812, 3821,
    3900, 3908, 3909
\l__enumext_item_width_viii_dim . . 3843, 3852,
    3949, 3957, 3958
\l__enumext_item_width_X_dim . . . . . 165
\l__enumext_itemindent_X_dim . . . . . 71
\l__enumext_itemsep_i_skip . . . 1216, 1223, 1226,
    1227, 1231, 1236, 1238, 1239, 1299, 1304
\l__enumext_itemsep_ii_skip . . . . . 1246, 1251
\l__enumext_itemsep_iii_skip . . . . . 1255, 1260
\l__enumext_itemsep_vii_skip . . . . . 4315
\l__enumext_itemsep_viii_skip . . . . . 4560
\l__enumext_joined_item_aux_vii_int . . 3894,
    3895, 3896, 3897, 3903
\l__enumext_joined_item_aux_viii_int . 3943,
    3944, 3945, 3946, 3952
\l__enumext_joined_item_aux_X_int . . . 165
\__enumext_joined_item_vii:w . . 109, 4167, 4168,
    4170, 4170
\l__enumext_joined_item_vii_int . . 3865, 3866,
    3869, 3871, 3877, 3882, 3887, 3892, 3894, 3900
\__enumext_joined_item_viii:w 114, 4391, 4392,
    4394, 4394
\l__enumext_joined_item_viii_int . 3914, 3915,
    3918, 3920, 3926, 3931, 3936, 3941, 3943, 3949
\l__enumext_joined_item_X_int . . . . . 165
\l__enumext_joined_width_vii_dim . 3898, 3905,
    3908, 4276, 4288
\l__enumext_joined_width_viii_dim 3947, 3954,
    3957, 4510, 4530
\l__enumext_joined_width_X_dim . . . . . 165
\__enumext_keyans_addto_prop:n 82, 2796, 2796,
    3174, 3670
\__enumext_keyans_addto_seq:n . 83, 2870, 2870,
    3176, 3672
\__enumext_keyans_addto_seq_link: 2870, 2891,
    2893, 4450
\__enumext_keyans_anspic_code:nnn . . 99, 100,
    3661, 3664, 3664
\__enumext_keyans_default_item:n . . 89, 3155,
    3155, 3191
\l__enumext_keyans_env_bool 34, 3387, 3400, 3554,
    3641
\__enumext_keyans_fake_item: . . 899, 915, 3262
\l__enumext_keyans_level_h_int . . 28, 642, 669,
    2515, 2577, 2848, 4102, 4344, 4345
\l__enumext_keyans_level_int . . 28, 1414, 2511,
    2573, 2843, 3553, 3558, 3655
\__enumext_keyans_make_label: 37, 90, 3179, 3195,
    3260
\__enumext_keyans_mini_right_cmd:n 55, 1416,
    1453, 1453
\__enumext_keyans_mini_set_vskip: . . . . . 52
\__enumext_keyans_minipage_add_space: . . 52,
    1314, 1314, 3583
\__enumext_keyans_minipage_set_skip: . 1279,
    1279, 1316
\__enumext_keyans_multi_addvspace: 1150, 1161,
    3607
\__enumext_keyans_multi_set_vskip: 49, 1150,
    1150, 1163
\__enumext_keyans_multicols_start: 3571, 3586,
    3588
\__enumext_keyans_multicols_stop: 1457, 3571,
    3613, 3639
\__enumext_keyans_name_and_start: 26, 32, 288,
    288, 3555, 3715, 4349
\__enumext_keyans_parse_keys:n 3514, 3567, 3567
\l__enumext_keyans_pic_above_int . 139, 3755,
    3756, 3758
\l__enumext_keyans_pic_above_skip . 101, 139,
    3694, 3734
\__enumext_keyans_pic_arg_two: 101, 3692, 3722,
    3722
\l__enumext_keyans_pic_below_int . 139, 3755,
    3756, 3759
\l__enumext_keyans_pic_body_seq . 99, 100, 102,
    139, 3659, 3699, 3763
\__enumext_keyans_pic_do:n 101, 3699, 3701, 3747,
    3747, 3751
\l__enumext_keyans_pic_level_int . . 28, 1398,
    2519, 2581, 2799, 2838, 2873, 2961, 3710, 3711
\__enumext_keyans_pic_row:n 101, 102, 3749, 3752,
    3752
\__enumext_keyans_pic_safe_exec: . 100, 3688,
    3708, 3708
\__enumext_keyans_pic_skip_abs:N . 101, 3717,
    3717, 3733
\l__enumext_keyans_pic_width_dim . 139, 3754,
    3761
\__enumext_keyans_redefine_item: . . 89, 3179,
    3179, 3259
\__enumext_keyans_ref: . . . . . 41, 694, 712, 3261
\__enumext_keyans_ref:n . . . . . 40, 691, 694, 694
\__enumext_keyans_safe_exec: . 3513, 3547, 3547
\__enumext_keyans_set_item_width: . 97, 3522,
    3530, 3530
\__enumext_keyans_show_ans: . . 2914, 2922, 2941
\__enumext_keyans_show_item_opt: . 2914, 2929,
    3172, 3684, 4516
\__enumext_keyans_show_left:n . 89, 2914, 2914,
    3170, 3679
\__enumext_keyans_show_pos: . . 2914, 2926, 2954
\__enumext_keyans_starred_item:n . . 89, 3167,
    3167, 3187
\__enumext_keyans_store_ref: . . 82, 2817, 2817,
    3175, 3671, 4448
\__enumext_keyans_store_ref_aux_i: 83, 2817,
    2829, 2832
\__enumext_keyans_store_ref_aux_ii: 83, 2817,
    2858, 2860
\__enumext_keyans_unknown_keys:n . 3093, 3097,
    3101
\__enumext_keyans_unknown_keys:nn 3093, 3103,
    3105
\__enumext_keyans_wrapper_opt:n . . 2065, 2937
\l__enumext_label_copy_i_tl . . 2323, 2836, 2841,
    2846, 2851
\l__enumext_label_copy_v_tl . . . . . 2846

```

`\l__enumext_label_copy_v_i_tl` . . . . . 2841  
`\l__enumext_label_copy_v_ii_tl` 2299, 2310, 2339, 2836  
`\l__enumext_label_copy_v_iii_tl` . . . . . 2851  
`\l__enumext_label_copy_X_tl` . . . . . 154  
`\l__enumext_label_fill_left_v_tl` . . . . . 3199  
`\l__enumext_label_fill_left_X_tl` . . . . . 98  
`\l__enumext_label_fill_right_v_tl` . . . . . 3206  
`\l__enumext_label_fill_right_X_tl` . . . . . 98  
`\l__enumext_label_font_style_v_tl` 3200, 3683  
`\l__enumext_label_font_style_v_ii_tl` . . . 4264  
`\l__enumext_label_font_style_v_iii_tl` . . 4498  
`\l__enumext_label_i_tl` . . . . . 557  
`\l__enumext_label_ii_tl` . . . . . 557  
`\l__enumext_label_iii_tl` . . . . . 557  
`\l__enumext_label_iv_tl` . . . . . 557  
`\__enumext_label_style:Nnn` 26, 36, 471, 471, 486, 562, 609, 680, 684  
`\l__enumext_label_v_tl` . . 82, 83, 677, 2804, 2878, 2948, 2988, 3169, 3173, 3517, 3678, 3680  
`\l__enumext_label_v_i_tl` . 82, 83, 677, 2801, 2875, 3678, 3680, 3684  
`\l__enumext_label_v_ii_tl` . 604, 4190, 4215, 4222  
`\l__enumext_label_v_iii_tl` 604, 4414, 4442, 4446  
`\l__enumext_label_width_by_box` . . 67, 467, 468  
`\__enumext_label_width_by_box:Nn` 36, 465, 465, 470, 482, 745  
`\l__enumext_labelsep_i_dim` . . 2946, 2951, 2959, 2991, 4454, 4469  
`\l__enumext_labelsep_v_dim` . . . . . 3597  
`\l__enumext_labelsep_v_ii_dim` . 2423, 2946, 2959, 3807, 3817, 3901, 4213, 4274, 4290  
`\l__enumext_labelsep_v_iii_dim` 3838, 3848, 3950, 4508, 4517, 4532  
`\l__enumext_labelwidth_i_dim` . 2945, 2951, 2958, 2991, 4454, 4469  
`\l__enumext_labelwidth_v_dim` . . . . . 3597  
`\l__enumext_labelwidth_v_ii_dim` . . 2423, 2945, 2958, 3807, 3816, 3901, 4267, 4271, 4289  
`\l__enumext_labelwidth_v_iii_dim` . . 3838, 3847, 3950, 4501, 4505, 4531  
`\l__enumext_leftmargin_tmp_v_bool` . 101, 3724  
`\l__enumext_leftmargin_tmp_X_bool` . . . . . 71  
`\l__enumext_leftmargin_tmp_X_dim` . . . . . 71  
`\l__enumext_leftmargin_X_dim` . . . . . 71  
`\__enumext_level:` 205, 205, 586, 589, 590, 599, 601, 902, 906, 910, 977, 981, 985, 989, 1098, 1100, 1102, 1104, 1137, 1139, 1141, 1143, 1148, 1183, 1189, 1194, 1195, 1201, 1204, 1275, 1423, 1427, 1436, 1501, 1503, 1505, 1508, 1515, 1517, 1519, 1522, 2117, 2119, 2121, 2149, 2150, 2152, 2208, 2216, 2220, 2224, 2427, 2428, 3003, 3004, 3008, 3009, 3010, 3018, 3026, 3027, 3030, 3037, 3038, 3042, 3045, 3047, 3065, 3066, 3067, 3070, 3073, 3323, 3325, 3344, 3349, 3393, 3406, 3413, 3424, 3426, 3429, 3430, 3432, 3436, 3443, 3446, 3448, 3450, 3451, 3452, 3453, 3456, 3461, 3467, 3473, 3476, 3480, 3484, 3490  
`\l__enumext_level_h_int` 107, 28, 238, 261, 275, 625, 662, 1405, 1933, 1953, 2318, 2551, 2563, 3401, 4097, 4098  
`\l__enumext_level_int` . 93, 28, 207, 248, 260, 276, 370, 1110, 1213, 1404, 1927, 1959, 2295, 2305, 2311, 2317, 2324, 2333, 2338, 2550, 2562, 2778, 3275, 3357, 3358, 3369, 3377, 3391, 3404, 3457, 3562, 3651, 4141, 4151, 4357, 5202, 5206, 5212, 5216  
`\__enumext_list_arg_two_i:` . . . . . 3241  
`\__enumext_list_arg_two_ii:` . . . . . 3241  
`\__enumext_list_arg_two_iii:` . . . . . 3241  
`\__enumext_list_arg_two_iv:` . . . . . 3241  
`\__enumext_list_arg_two_v:` . 89, 3241, 3519, 3725  
`\__enumext_list_arg_two_vii:` . . . . . 3281, 4078  
`\__enumext_list_arg_two_viii:` . . . . . 3281, 4327  
`\l__enumext_listoffset_v_dim` . 3538, 3543, 3599  
`\l__enumext_listparindent_v_ii_dim` . . . . 4277  
`\l__enumext_listparindent_v_iii_dim` . . . 4511  
`\__enumext_log_answer_vars:` . 33, 342, 350, 2785  
`\__enumext_log_global_vars:` . 33, 342, 342, 2784  
`\__enumext_make_label` . . . . . 3050  
`\__enumext_make_label:` . . . . . 37, 87, 3061, 3270  
`\l__enumext_mark_answer_sym_tl` 72, 2071, 2274, 2440, 2963, 2976, 4458  
`\l__enumext_mark_position_str` 128, 2075, 2076, 2102, 2103, 2272  
`\l__enumext_mark_ref_sym_tl` . . 2088, 2413, 2905  
`\l__enumext_meta_path_tl` . 124, 4790, 4791, 4793, 4794  
`\c__enumext_meta_paths_prop` . . . . . 121, 4766  
`\__enumext_mini_addvspace_v_ii:` 54, 1384, 1384, 3975  
`\__enumext_mini_addvspace_v_iii:` 54, 1384, 1390, 4030  
`\__enumext_mini_env*` . . . . . 368  
`\__enumext_mini_right_cmd:n` . 54, 55, 1418, 1420, 1420  
`\__enumext_mini_set_vskip_v_ii:` 53, 1327, 1327, 1386  
`\__enumext_mini_set_vskip_v_iii:` 53, 1327, 1349, 1392  
`\__enumext_minipage:w` 33, 363, 365, 374, 3761, 4276, 4510  
`\l__enumext_minipage_active_v_bool` 3581, 3604, 3618, 3626  
`\g__enumext_minipage_active_v_ii_bool` . . 105, 3986, 3991, 4010  
`\l__enumext_minipage_active_v_ii_bool` . 3971, 3982  
`\g__enumext_minipage_active_v_iii_bool` 4041, 4046, 4065  
`\l__enumext_minipage_active_v_iii_bool` 4026, 4037  
`\g__enumext_minipage_active_X_bool` . . . 165  
`\l__enumext_minipage_active_X_bool` . . . . 86  
`\__enumext_minipage_add_space:` . . 51, 95, 1179, 1265, 3434  
`\g__enumext_minipage_after_skip` 86, 1331, 1343, 4008, 4063  
`\l__enumext_minipage_after_skip` . . 50, 96, 86, 1192, 1193, 1196, 1216, 1219, 1223, 1226, 1228, 1231, 1236, 1238, 1240, 1248, 1251, 1257, 1260, 1281, 1295, 1297, 1301, 1304, 1351, 1364, 1378, 1429, 1431, 1460, 3636  
`\g__enumext_minipage_center_v_ii_bool` . 3995, 4011  
`\g__enumext_minipage_center_v_iii_bool` 4050, 4066  
`\g__enumext_minipage_center_X_bool` . . . 165  
`\l__enumext_minipage_hsep_v_dim` . . . . . 3579  
`\l__enumext_minipage_hsep_v_ii_dim` . . . . 3969

```

\l__enumext_minipage_hsep_viii_dim ... 4024
\l__enumext_minipage_left_skip 86, 1205, 1208,
    1277, 1282, 1329, 1334, 1338, 1352, 1356, 1370, 1388,
    1394
\l__enumext_minipage_left_v_dim .. 3577, 3584
\l__enumext_minipage_left_vii_dim 3965, 3977
\l__enumext_minipage_left_viii_dim 4020, 4032
\l__enumext_minipage_left_X_dim ..... 86
\g__enumext_minipage_right_skip 86, 1330, 1335,
    1339, 3994, 4049
\l__enumext_minipage_right_skip . 50, 86, 1181,
    1187, 1192, 1193, 1194, 1195, 1283, 1284, 1290, 1295,
    1311, 1353, 1360, 1374, 1439, 1467
\l__enumext_minipage_right_v_dim . 1455, 1464,
    3575, 3579
\g__enumext_minipage_right_vii_dim 105, 3973,
    3993, 4013
\l__enumext_minipage_right_vii_dim 105, 3963,
    3968, 3974
\g__enumext_minipage_right_viii_dim .. 4028,
    4048, 4068
\l__enumext_minipage_right_viii_dim .. 4018,
    4023, 4029
\g__enumext_minipage_right_X_dim ..... 165
\g__enumext_minipage_right_X_skip .... 165
\__enumext_minipage_set_skip: . 50, 1179, 1179,
    1267
\g__enumext_minipage_stat_int 95, 86, 1444, 1472,
    3433, 3492, 3497, 3582, 3628, 3633
\l__enumext_miniright_code_vii_box 4002, 4006
\g__enumext_miniright_code_vii_tl 106, 3997,
    4004, 4012
\l__enumext_miniright_code_viii_box .. 4057,
    4061
\g__enumext_miniright_code_viii_tl 4052, 4059,
    4067
\l__enumext_miniright_code_X_box ..... 165
\__enumext_multi_addvspace: . 49, 95, 1132, 1132,
    3464
\__enumext_multi_set_vskip: 48, 1070, 1096, 1134
\l__enumext_multicols_above_ii_skip ... 1115
\l__enumext_multicols_above_iii_skip .. 1121
\l__enumext_multicols_above_iv_skip ... 1127
\l__enumext_multicols_above_v_skip 1152, 1166,
    1177
\l__enumext_multicols_above_X_skip .... 79
\l__enumext_multicols_below_ii_skip .. 1220,
    1227, 1229, 1239, 1241
\l__enumext_multicols_below_v_skip 1156, 1170,
    3620
\l__enumext_multicols_below_X_skip .... 79
\g__enumext_multicols_right_X_skip .... 79
\__enumext_multicols_start: 95, 3438, 3440, 3440
\__enumext_multicols_stop: 96, 1425, 3470, 3470,
    3502
\__enumext_nested_base_line_fix: . 43, 93, 108,
    819, 829, 3373, 4118
\__enumext_newlabel:nn 29, 35, 73, 422, 422, 2349,
    2864
\l__enumext_newlabel_arg_one_tl 29, 35, 73, 83,
    154, 2342, 2350, 2412, 2853, 2865, 2903
\l__enumext_newlabel_arg_two_tl 29, 35, 72, 154,
    2298, 2308, 2321, 2336, 2351, 2840, 2845, 2850, 2866
\__enumext_parse_foreach_keys:n .. 4815, 4831,
    4848
\__enumext_parse_foreach_keys:nn . 4815, 4838,
    4850
\__enumext_parse_keys:n 43, 59, 3319, 3364, 3364
\__enumext_parse_keys_vii:n . 43, 59, 4073, 4110,
    4110
\__enumext_parse_keys_viii:n . 4323, 4362, 4362
\__enumext_parse_save_key:n 69, 2142, 2147, 2147
\__enumext_parse_save_key_vii:n 69, 2137, 2147,
    2155
\__enumext_parse_series:n 59, 93, 108, 1630, 1630,
    3372, 4116
\__enumext_parse_store_keys:n ..... 93
\l__enumext_parsep_i_skip ..... 1113, 1115
\l__enumext_parsep_ii_skip ..... 1119, 1121
\l__enumext_parsep_iii_skip ..... 1125, 1127
\l__enumext_parsep_vii_skip ..... 4278
\l__enumext_parsep_viii_skip ..... 4512
\l__enumext_partopsep_v_skip . 1168, 1172, 1292,
    1323
\l__enumext_partopsep_viii_skip ..... 1362
\__enumext_phantomsection: 35, 386, 415, 419, 435
\__enumext_previus_level_skip: ... 1198, 1211
\__enumext_print_footnote: ... 3768, 3791, 4296,
    4538
\__enumext_print_keyans_box:NN 72, 2266, 2266,
    2279, 2423, 2426, 2950, 2990, 4454, 4469
\l__enumext_print_keyans_i_tl ... 4600, 4622
\l__enumext_print_keyans_ii_tl ... 4604, 4623
\l__enumext_print_keyans_iii_tl .. 4608, 4624
\l__enumext_print_keyans_iv_tl ... 4612, 4625
\l__enumext_print_keyans_starred_tl 118, 119,
    128, 4596, 4644
\l__enumext_print_keyans_vii_tl 118, 4616, 4626
\l__enumext_print_keyans_X_tl ..... 128
\__enumext_printkeyans:nnn 118, 119, 4627, 4630,
    4630
\__enumext_redefine_item: . 87, 3050, 3050, 3269
\l__enumext_ref_key_arg_tl 38, 50, 220, 579, 580,
    593, 624, 627, 638, 644, 655, 696, 697, 708
\l__enumext_ref_the_count_tl . 38, 50, 586, 589,
    592, 632, 634, 637, 649, 651, 654, 702, 704, 707
\__enumext_regex_counter_style: .. 30, 38, 215,
    215, 587, 633, 650, 703
\__enumext_register_counter_style:Nn .. 455,
    455, 460, 461, 462, 463, 464
\__enumext_remove_extra_parsep_vii: .. 4088,
    4305, 4305
\__enumext_remove_extra_parsep_viii: . 4337,
    4548, 4548
\__enumext_renew_footnote: ... 3768, 3772, 4248,
    4491
\l__enumext_renew_the_count_v_tl 705, 714, 716
\l__enumext_renew_the_count_vii_tl 635, 664,
    666
\l__enumext_renew_the_count_viii_tl 652, 671,
    673
\l__enumext_renew_the_count_X_tl ..... 50
\__enumext_rescan_anskey_env:n .. 79, 81, 2660,
    2755, 2763
\__enumext_reset_global_bool: .. 318, 321, 330
\__enumext_reset_global_int: ... 318, 320, 324
\__enumext_reset_global_tl: .... 318, 322, 336
\__enumext_reset_global_vars: . 32, 81, 318, 318,

```

```

2793
\l__enumext_resume_active_bool 59, 61, 61, 1634,
1754
\__enumext_resume_counter: . . 60, 61, 1752, 1758,
1765
\__enumext_resume_counter:n . 59, 61, 1723, 1728,
1752, 1752, 1822, 1830
\__enumext_resume_counter_save_ans: . . 61, 62,
1752, 1763, 1795
\__enumext_resume_counter_series: . 61, 1752,
1761, 1778
\g__enumext_resume_int . . . 61, 1675, 1769, 1770
\__enumext_resume_last:n . . 59, 1630, 1636, 1649
\l__enumext_resume_name_tl 61, 1671, 1679, 1682,
1698, 1706, 1709, 1755, 1756, 1784, 1791
\__enumext_resume_save_counter: . . 59, 96, 108,
1662, 1662, 3508, 4135
\__enumext_resume_series:n . 60, 1598, 1719, 1719
\__enumext_resume_starred: . 62, 1599, 1816, 1816
\g__enumext_resume_vii_int 61, 1702, 1774, 1775
\l__enumext_rightmargin_vii_dim . . 3819, 3823,
3828
\l__enumext_rightmargin_viii_dim . 3850, 3854,
3859
\__enumext_safe_exec: . . 34, 93, 3318, 3353, 3353
\__enumext_safe_exec_vii: . 34, 4072, 4093, 4093
\__enumext_safe_exec_viii: . . . 4322, 4342, 4342
\l__enumext_series_name_tl . . . . . 61
\l__enumext_series_str . . 59, 93, 108, 1596, 1632,
1640, 1641, 1643, 1645, 1666, 1669, 1673, 1693, 1696,
1700, 3368, 4114
\__enumext_set_error:nn . . . . . 4752, 4762, 4764
\__enumext_set_item_width: . 93, 3328, 3336, 3336
\__enumext_set_parse:n . . . . . 4736, 4752, 4752
\l__enumext_setkey_tmpa_int . . . 119, 4729, 4733
\l__enumext_setkey_tmpa_seq . . 119, 4727, 4737,
4743, 4745, 4747, 4759
\l__enumext_setkey_tmpa_tl . . . 119, 4735, 4739
\l__enumext_setkey_tmpb_seq . . 119, 4728, 4731,
4735, 4736
\l__enumext_setkey_tmpb_tl 119, 4754, 4756, 4757
\l__enumext_show_answer_bool . 2082, 2106, 2434,
2920, 2934, 3675, 4452
\__enumext_show_length:nnn . . 45, 223, 223, 4973,
4974, 4975, 4976, 4977, 4978, 4979, 4980, 4981, 4982,
4988, 4989, 4990, 4991, 4992, 4993, 4994, 4995, 4996,
4997
\l__enumext_show_position_bool . . . 2085, 2109,
2438, 2924, 2935, 3676, 4456
\g__enumext_standar_bool 31, 93, 34, 237, 240, 259,
333, 1664, 1729, 1741, 1767, 1780, 1818, 1958, 1972,
2303, 2316, 2331, 3388
\l__enumext_standar_bool . 93, 96, 34, 2304, 3360,
3507, 4107
\l__enumext_standar_first_bool 31, 93, 34, 264,
832, 1651, 1798, 1860, 1867
\__enumext_standar_item_vii:w . 109, 110, 4175,
4177, 4177
\__enumext_standar_item_viii:w 114, 4399, 4401,
4401
\__enumext_standar_ref: . . . . 39, 577, 597, 3271
\__enumext_standar_ref:n . . . . 38, 569, 577, 577
\g__enumext_standar_series_tl . 61, 1653, 1654,
1820, 1823
\__enumext_standar_unknown_keys:n 3133, 3137,
3141
\__enumext_standar_unknown_keys:nn 3133, 3143,
3145
\g__enumext_starred_bool 31, 107, 34, 247, 250, 274,
334, 1691, 1734, 1745, 1772, 1787, 1826, 1932, 1978,
2294, 2834, 4014
\l__enumext_starred_bool 107, 108, 34, 1410, 2332,
2367, 2373, 2421, 2709, 2714, 2943, 2956, 3361, 4106,
4134, 4350, 4354
\__enumext_starred_columns_set_vii: . . 3801,
3801, 4081
\__enumext_starred_columns_set_viii: . 3801,
3832, 4330
\l__enumext_starred_first_bool 31, 107, 34, 279,
843, 1656, 1807, 1860, 1867
\__enumext_starred_item:nn . . . 3013, 3013, 3056
\__enumext_starred_item_exec: . 115, 4444, 4444,
4495
\__enumext_starred_item_vii:w . 109, 110, 4174,
4193, 4193
\__enumext_starred_item_vii_aux_i:w . . 4193,
4198, 4201
\__enumext_starred_item_vii_aux_ii:w . 4193,
4199, 4204, 4206
\__enumext_starred_item_vii_aux_iii:w 4193,
4209, 4218
\__enumext_starred_item_viii:w 114, 115, 4398,
4417, 4417
\__enumext_starred_item_viii_aux_i:w . . 115,
4417, 4422, 4425
\__enumext_starred_item_viii_aux_ii:w . 115,
4417, 4423, 4437, 4439
\__enumext_starred_joined_item_vii:n 104, 109,
3863, 3863, 4172
\__enumext_starred_joined_item_viii:n . 104,
114, 3863, 3912, 4396
\__enumext_starred_ref: . . . . 40, 622, 660, 3301
\__enumext_starred_ref:n . . . . 39, 616, 622, 622
\g__enumext_starred_series_tl . 61, 1658, 1659,
1828, 1831
\__enumext_starred_unknown_keys:n 3115, 3117,
3119
\__enumext_starred_unknown_keys:nn 3115, 3121,
3123
\__enumext_start_from:NNn 41, 719, 719, 732, 754,
760
\l__enumext_start_i_int . . . . 1770, 1782, 1801
\__enumext_start_item_tmp_vii: 107, 4084, 4157,
4157
\__enumext_start_item_tmp_viii: . . 112, 4333,
4381, 4381
\__enumext_start_item_vii:w . . . 110, 4185, 4190,
4215, 4222, 4224, 4224
\__enumext_start_item_viii:w . 114, 4409, 4414,
4442, 4472, 4472
\g__enumext_start_line_tl 31, 34, 267, 282, 339,
2002, 2007, 2012, 2026, 2031, 2036
\__enumext_start_list:nn . . 33, 90, 101, 357, 359,
3322, 3516, 3689, 4076, 4325
\__enumext_start_mini_vii: 108, 3961, 3961, 4126
\__enumext_start_mini_viii: . . 113, 4016, 4016,
4373
\__enumext_start_save_ans_msg: 63, 1844, 1844,

```

```

1869
\__enumext_start_store_level: . 94, 3321, 3382,
3382
\__enumext_start_store_level_vii: 109, 4075,
4137, 4137
\l__enumext_start_vii_int ... 1775, 1789, 1810
\l__enumext_start_X_int ..... 98
\__enumext_stop_item_tmp_vii: .. 107, 109, 110,
4083, 4087, 4159, 4226
\__enumext_stop_item_tmp_viii: 112, 114, 4332,
4336, 4383, 4474
\__enumext_stop_item_vii: 110, 112, 4226, 4281,
4281
\__enumext_stop_item_viii: 116, 4474, 4523, 4523
\__enumext_stop_list: .. 33, 357, 360, 3332, 3527,
3702, 4089, 4339
\__enumext_stop_mini_vii: 105, 108, 3961, 3980,
4130
\__enumext_stop_mini_viii: 113, 4016, 4035, 4377
\__enumext_stop_save_ans_msg: . 63, 1844, 1849,
2782
\__enumext_stop_store_level: .. 94, 3333, 3382,
3411
\__enumext_stop_store_level_vii: . 109, 4090,
4137, 4147
\l__enumext_store_active_bool 28, 63, 110, 1799,
1808, 1876, 2507, 3386, 3399, 3549, 3557, 3647, 3706,
4139, 4149, 4356
\__enumext_store_active_keys:n 68, 69, 93, 2115,
2115, 3379
\__enumext_store_active_keys_vii:n 68, 69, 108,
2115, 2125, 4117
\__enumext_store_addto_prop:n 70, 82, 2190, 2190,
2198, 2358, 2815, 4447
\__enumext_store_addto_seq:n 70, 84, 2199, 2199,
2203, 2210, 2224, 2232, 2241, 2255, 2263, 2416, 2908
\l__enumext_store_anskey_arg_tl 28, 73, 74, 110,
2364, 2369, 2371, 2376, 2383, 2386, 2396, 2401, 2404,
2410, 2416
\__enumext_store_anskey_code:n 73, 76, 81, 2355,
2355, 2500, 2753, 2761
\l__enumext_store_anskey_env_tl .. 28, 79, 110,
2683, 2687, 2693, 2755, 2763
\l__enumext_store_anskey_opt_tl .. 28, 80, 110,
2684, 2711, 2717, 2724, 2730, 2740, 2750, 2759
\__enumext_store_anskey_safe_outer: .... 76
\g__enumext_store_columns_break_bool . 2607,
2708, 2770
\l__enumext_store_columns_break_bool . 2366,
2456
\l__enumext_store_current_label_tl 28, 82-84,
115, 110, 2798, 2801, 2804, 2811, 2813, 2815, 2872,
2875, 2878, 2884, 2889, 2899, 2908, 4427, 4432, 4433,
4446, 4447, 4449
\l__enumext_store_current_label_tmp_tl . 28,
110, 3169, 3173
\l__enumext_store_current_opt_arg_tl 28, 115,
110, 2918, 2931, 2937, 4435
\__enumext_store_internal_ref: .. 72, 73, 2280,
2280, 2361
\g__enumext_store_item_join_int .. 2610, 2715,
2719, 2771
\l__enumext_store_item_join_int .. 2374, 2378,
2459
\g__enumext_store_item_star_bool . 2612, 2722,
2772
\l__enumext_store_item_star_bool . 2381, 2461
\g__enumext_store_item_symbol_sep_dim 2617,
2737, 2742, 2774
\l__enumext_store_item_symbol_sep_dim 2393,
2398, 2466
\g__enumext_store_item_symbol_tl . 2615, 2728,
2732, 2773
\l__enumext_store_item_symbol_tl . 2384, 2388,
2464
\l__enumext_store_keyans_item_opt_sep_-
tl .... 2068, 2809, 2811, 2882, 2886, 4430, 4432
\__enumext_store_level_close: . 71, 2204, 2228,
3415
\__enumext_store_level_close_vii: . 71, 2235,
2259, 4153
\__enumext_store_level_open: 71, 94, 2204, 2204,
3394, 3407
\__enumext_store_level_open_vii: .. 71, 2235,
2235, 4143
\g__enumext_store_name_tl 28, 63, 110, 338, 345,
346, 347, 348, 1852, 1878, 2001, 2006, 2011, 2025,
2030, 2035, 2780
\l__enumext_store_name_tl 28, 63, 65, 110, 1685,
1688, 1712, 1715, 1803, 1812, 1847, 1856, 1857, 1878,
1879, 1880, 1882, 1883, 1885, 1887, 1888, 1890, 1892,
1893, 1917, 2192, 2194, 2201, 2344, 2345, 2446, 2689,
2855, 2856, 2969, 2982, 4464
\l__enumext_store_ref_key_bool 73, 2091, 2359,
2407, 2819, 2896
\l__enumext_store_save_key_vii_bool .. 2127,
2157
\l__enumext_store_save_key_vii_tl 2129, 2130,
2158, 2159, 2239, 2247, 2251, 2255
\l__enumext_store_save_key_X_bool .. 68, 128
\l__enumext_store_save_key_X_tl .. 68, 69, 128
\l__enumext_store_upper_level_X_bool .. 128
\__enumext_storing_exec: 63, 78, 1854, 1870, 1874
\__enumext_storing_set:n .. 63, 1839, 1854, 1854
\l__enumext_the_counter_v_tl ..... 704
\l__enumext_the_counter_vii_tl ..... 634
\l__enumext_the_counter_viii_tl ..... 651
\l__enumext_the_counter_X_tl ..... 50
\__enumext_tmp:n 45, 49, 54, 60, 71, 78, 79, 85, 92, 97,
98, 109, 131, 138, 157, 161, 165, 185, 819, 828, 1592,
1603, 1835, 1843, 1896, 1914, 2055, 2096, 2097, 2114,
2133, 2146, 2282, 2289, 2290, 2311, 2324, 2327, 2338,
2821, 2828, 3093, 3100, 3133, 3140, 3241, 3280, 3281,
3315
\__enumext_tmp:nn 487, 508, 509, 540, 541, 556, 749,
774, 855, 877, 878, 898, 951, 959, 960, 974, 1039, 1055,
1056, 1069, 1481, 1497, 3077, 3092
\__enumext_tmp:nnn 557, 573, 574, 575, 576, 604, 620,
621
\__enumext_tmp:nnnnn 775, 800, 803, 806, 808, 810,
813, 816
\__enumext_tmp:w ..... 4575, 4578
\l__enumext_tmpa_vii_int 3811, 3814, 3823, 3854
\l__enumext_tmpa_viii_int ..... 3842, 3845
\l__enumext_tmpa_X_dim ..... 165
\l__enumext_tmpa_X_int ..... 165
\l__enumext_topsep_v_skip 1154, 1158, 1286, 3705,
3737
\l__enumext_topsep_vii_skip .. 1332, 1341, 1345
\l__enumext_topsep_viii_skip . 1354, 1376, 1380

```



`\__enumext_undefine_anskey_env`: . 77, 81, [2540](#),  
2540, 2791

`\__enumext_undo_space`: .. 1070, 1146, 1175, 1268,  
1430, 3478, 3479, 3483, 3498

`\l__enumext_vspace_a_star_v_bool` ..... 1530

`\l__enumext_vspace_a_star_vii_bool` ... 1552

`\l__enumext_vspace_a_star_viii_bool` ... 1563

`\l__enumext_vspace_a_star_X_bool` ..... [98](#)

`\__enumext_vspace_above`: 56, 94, [1498](#), 1498, 3420

`\__enumext_vspace_above_v`: . 56, [1526](#), 1526, 3573

`\l__enumext_vspace_above_v_skip` .. 1528, 1532,  
1534

`\__enumext_vspace_above_vii`: 57, 108, [1548](#), 1548,  
4123

`\l__enumext_vspace_above_vii_skip` 1550, 1554,  
1556

`\__enumext_vspace_above_viii`: . 57, [1548](#), 1559,  
4371

`\l__enumext_vspace_above_viii_skip` 1561, 1565,  
1567

`\l__enumext_vspace_b_star_v_bool` ..... 1541

`\l__enumext_vspace_b_star_vii_bool` ... 1574

`\l__enumext_vspace_b_star_viii_bool` ... 1585

`\l__enumext_vspace_b_star_X_bool` ..... [98](#)

`\__enumext_vspace_below`: 56, 96, [1512](#), 1512, 3506

`\__enumext_vspace_below_v`: . 57, [1537](#), 1537, 3643

`\l__enumext_vspace_below_v_skip` .. 1539, 1543,  
1545

`\__enumext_vspace_below_vii`: 57, 108, [1570](#), 1570,  
4133

`\l__enumext_vspace_below_vii_skip` 1572, 1576,  
1578

`\__enumext_vspace_below_viii`: . 57, [1570](#), 1581,  
4379

`\l__enumext_vspace_below_viii_skip` 1583, 1587,  
1589

`\__enumext_widest_from:nNNn` .. 41, [733](#), 733, 748,  
767

`\g__enumext_widest_label_tl` 26, 36, [67](#), 475, 479,  
483

`\l__enumext_wrap_label_opt_v_bool` .... 3163

`\l__enumext_wrap_label_opt_vii_bool` [110](#), 4184

`\l__enumext_wrap_label_opt_viii_bool` .. [114](#),  
4408

`\l__enumext_wrap_label_opt_X_bool` ..... [98](#)

`\l__enumext_wrap_label_v_bool` 3159, 3163, 3171,  
3201

`\l__enumext_wrap_label_vii_bool` .. [110](#), 4183,  
4188, 4196, 4265

`\l__enumext_wrap_label_viii_bool` . [114](#), 4407,  
4412, 4420, 4499

`\l__enumext_wrap_label_X_bool` ..... [98](#)

`\__enumext_wrapper_label_v:n` ..... 3203, 3684

`\__enumext_wrapper_label_vii:n` ..... 4268

`\__enumext_wrapper_label_viii:n` ..... 4502

`\l__enumext_write_aux_file_tl` . 29, 73, 83, [154](#),  
2347, 2353, 2862, 2868

`enumext*` ..... 5, [4070](#)

`enumXi` ..... [447](#)

`enumXii` ..... [447](#)

`enumXiii` ..... [447](#)

`enumXiv` ..... [447](#)

`enumXv` ..... [447](#)

`enumXvi` ..... [447](#)

`enumXvii` ..... [447](#)

`enumXviii` ..... [447](#)

Environments provide by `enumext`:

`anskey*` 28, 63, 72–75, 77–79, 81, 94, 109, 118, 124, 126

`enumext*` 25, 26, 29–31, 34, 36, 39–45, 47, 53, 54, 57–60,  
62–65, 67–77, 80, 82, 83, 87, 88, 92–94, 102–104, 106,  
109, 110, 112, 114, 116, 118, 119, 121, 125, 128, 129

`enumext` 25, 26, 30, 31, 34, 36–43, 45–50, 52, 54–56, 58–60,  
62–65, 67–73, 75–77, 80, 82, 83, 85–88, 90–92, 94,  
96–98, 100, 103, 105, 107, 109, 118, 119, 121, 125, 126,  
128

`keyans*` 25, 26, 28–32, 36, 39–45, 47, 53, 54, 57, 63, 64, 67,  
68, 70, 78, 82, 88, 92, 102–104, 113, 125, 127, 129

`keyanspic` 25, 26, 28, 29, 32, 36, 37, 40, 63, 64, 67, 70, 78,  
82–84, 99–101, 127

`keyans` 25, 26, 28, 29, 31, 32, 36, 37, 40–43, 45, 47, 49, 52,  
54–57, 63, 64, 67, 68, 70, 78, 82–84, 88–91, 96–101, 105,  
114, 125, 127

Environments:

`list` ..... 30, 33, 90, 92

`lrbox` ..... 103, 111, 112, 116

`minipage` .... 30, 33, 34, 47, 50, 99–103, 111, 112, 116

`multicols` ..... 47–50, 54, 95, 96

`scontents` ..... 78, 80

exp commands:

`\exp_after:wN` ..... 4578

`\exp_args:Ne` ..... 2752, 2760, 3376, 4566

`\exp_args:NV` ... 2472, 2627, 3103, 3121, 3143, 4850

`\exp_not:N` . 58, 478, 592, 637, 654, 707, 908, 922, 923,  
934, 935, 946, 947, 2412, 2443, 2444, 2901, 2966, 2967,  
2979, 2980, 4461, 4462, 4575

`\exp_not:n` 269, 284, 297, 305, 313, 531, 551, 592, 593,  
637, 638, 654, 655, 707, 708, 909, 1619, 1628, 2079,  
2176, 2188, 2350, 2378, 2388, 2398, 2412, 2413, 2719,  
2732, 2742, 2865, 2903, 2905, 4679, 4689, 4882, 4887

F

`\fbox` ..... 2062

`\fboxrule` ..... 2062

`\fboxsep` ..... 2062

file commands:

`\file_input_stop`: ..... 5286

`first` ..... [960](#)

`font` ..... [487](#)

`\footnote` ..... 102

`\footnote` ..... 102, 3776

`\footnotemark` ..... 3786

`\footnotesize` ..... 2444, 2967, 2980, 4462

`\footnotetext` ..... 3770

`\foreachkeyans` ..... 16, 122, [4815](#)

G

`\getkeyans` ..... 16, 117, [4564](#)

group commands:

`\group_begin`: .. 2442, 2487, 2662, 2749, 2965, 2978,  
4244, 4263, 4460, 4487, 4497, 4621

`\group_end`: 2449, 2503, 2766, 2972, 2985, 4273, 4285,  
4467, 4507, 4527, 4628

H

`\hbadness` ..... 4292, 4534

hbox commands:

`\hbox_set:Nn` ..... 467

`\hfill` 517, 521, 526, 527, 1434, 1463, 2412, 2901, 3985, 4040

hook commands:

`\hook_gput_code:nnn` ..... 9, 195, 199, 203, 384

\hook\_gremove\_code:nn . . . . . 80, 2678  
\hook\_gset\_rule:nnnn . . . . . 385  
\hook\_if\_empty:nTF . . . . . 2676  
\hspace . . . . . 4303, 4546  
\hyperlink . . . . . 74, 84  
\hyperlink . . . . . 2412, 2901  
\hypertarget . . . . . 35  
\hypertarget . . . . . 414

I

\IfHyperBoolean . . . . . 392  
\IfPackageLoadedTF . . . . . 11, 19, 388, 402  
\ignorespaces . . . . . 911  
\inputlineno . . . . . 269, 284, 297, 305, 313  
int commands:

\int\_add:Nn . . . . . 3896, 3945  
\int\_case:nn . . . 1110, 1213, 1927, 1953, 1992, 2016  
\int\_case:nnTF . . . . . 1072  
\int\_compare:nNnTF . . 370, 625, 642, 662, 669, 1200,  
1308, 1398, 1414, 1426, 1458, 2040, 2046, 2511, 2515,  
2519, 2527, 2573, 2577, 2581, 2778, 2799, 2838, 2843,  
2848, 2873, 2961, 3358, 3369, 3391, 3404, 3442, 3457,  
3472, 3492, 3558, 3562, 3590, 3615, 3628, 3651, 3655,  
3711, 3866, 3876, 3892, 3915, 3925, 3941, 4098, 4102,  
4141, 4151, 4298, 4307, 4345, 4357, 4540, 4550, 4733,  
4865  
\int\_compare\_p:nNn . . . 238, 248, 260, 261, 275, 276,  
1404, 1405, 1933, 1959, 2295, 2305, 2317, 2318, 2333,  
2374, 2550, 2551, 2562, 2563, 2715, 3401  
\int\_decr:N . . . . . 3895, 3944  
\int\_eval:n . . . 355, 762, 2194, 2345, 2444, 2856, 2967,  
2980, 3256, 3300, 3884, 3933, 4462  
\int\_from\_alph:n . . . . . 727, 741  
\int\_from\_roman:n . . . . . 729, 743  
\int\_gadd:Nn . . . . . 3897, 3946  
\int\_gdecr:N . . . . . 1936, 1941, 1945, 1949, 1962  
\int\_gincr:N 1769, 1774, 2357, 2911, 3000, 3034, 3177,  
3433, 3582, 3673, 4161, 4239, 4385, 4451  
\int\_gset:Nn . . . . . 1985, 3784  
\int\_gset\_eq:NN 1668, 1675, 1681, 1687, 1695, 1702,  
1708, 1714, 3781  
\int\_gzero:N . . . 326, 327, 328, 1444, 1472, 2052, 2771,  
3497, 3633, 4316, 4561  
\int\_if\_exist:NnTF 1643, 1679, 1685, 1706, 1712, 1890  
\int\_incr:N 2526, 3357, 3553, 3710, 4097, 4160, 4344,  
4384  
\int\_mod:nn . . . . . 4309, 4552  
\int\_new:N . . . 28, 29, 30, 31, 32, 33, 61, 62, 86, 102, 121,  
141, 142, 147, 148, 149, 151, 162, 168, 169, 170, 171,  
172, 1645, 1893  
\int\_set:Nn 723, 727, 729, 1782, 1789, 1801, 1810, 2663,  
3755, 3756, 3811, 3842, 3865, 3871, 3887, 3914, 3920,  
3936, 4292, 4534, 4729, 4867  
\int\_set\_eq:NN . . . . . 1770, 1775, 3894, 3943  
\int\_sign:n . . . . . 1987  
\int\_step\_function:nnN . . . . . 2311, 2324, 2338  
\int\_step\_function:nnnN . . . . . 4871  
\int\_step\_inline:nn . . . . . 4781  
\int\_step\_inline:nnn . . . . . 3757  
\int\_to\_roman:n . . . . . 207, 2291, 2328  
\int\_use:N 348, 353, 354, 1201, 1427, 1784, 1791, 1803,  
1812, 3256, 3275, 3300, 3377, 3443, 3452, 3467, 3473,  
3869, 3870, 3882, 3918, 3919, 3931, 5202, 5206, 5212,  
5216  
\int\_zero:N . . . . . 4301, 4544

\item . . . . . 85, 89, 109, 110, 114, 116, 361, 2212, 2218, 2243, 2249,  
2371, 2875, 2878, 3052, 3181, 3742, 4082, 4084, 4331,  
4333, 4449  
\item\* . . . . . 5, 14, 67, 3179  
item-pos\* . . . . . 3077  
item-sym\* . . . . . 3077  
\itemindent . . . . . 91  
\itemindent . . . . . 90  
itemindent . . . . . 855  
\itemsep . . . . . 100, 101  
\itemsep . . . . . 3726, 3732  
\itemwidth . . . . . 437, 2062, 3338, 3347, 3532, 3541, 3905, 3909,  
3954, 3958

K

keyans . . . . . 14, 3511  
keyans\* . . . . . 14, 4320  
keyanspic . . . . . 15, 3686

Keys for \anskey provide by enumext:

break-col . . . . . 73, 75, 78–80  
item-join . . . . . 74, 75, 78–80  
item-pos\* . . . . . 74, 75, 78–80  
item-star . . . . . 74, 75, 78–80  
item-sym\* . . . . . 74, 75, 78–80

Keys for anskey\* provide by enumext:

break-col . . . . . 73, 75, 78–80  
item-join . . . . . 74, 75, 78–80  
item-pos\* . . . . . 74, 75, 78–80  
item-star . . . . . 74, 75, 78–80  
item-sym\* . . . . . 74, 75, 78–80

Keys for environments provide by enumext:

above\* . . . . . 27, 56, 57, 94, 108  
above . . . . . 27, 56, 57, 94, 108, 113  
after . . . . . 45, 46, 96, 108, 113  
align . . . . . 27, 37, 87, 90, 111, 124  
base-fix . . . . . 43, 58, 70, 93, 108, 119  
before\* . . . . . 45, 46, 94, 108, 113  
before . . . . . 45, 46  
below\* . . . . . 27, 56, 57, 96, 108  
below . . . . . 27, 56, 57, 96, 108, 113  
check-ans 29–31, 62–64, 66, 67, 70, 81, 84, 96, 108, 112,  
125  
columns-sep . . . . . 47, 95  
columns . . . . . 27, 47, 55, 95  
first . . . . . 45, 46, 111  
font . . . . . 37, 87, 90, 111  
item-pos\* . . . . . 86, 87  
item-sym\* . . . . . 28, 86, 87  
itemindent . . . . . 27, 43, 44, 85, 86, 89, 111  
itemsep . . . . . 42, 92  
labeledsep . . . . . 37, 91, 111  
labelwidth . . . . . 36–41, 91  
label . . . . . 26, 36, 38, 41, 103  
lisparindent . . . . . 92  
list-indent . . . . . 27, 43, 44, 101  
list-offset . . . . . 43, 44, 93, 97  
listparindent . . . . . 43, 111  
mark-ans . . . . . 67, 70, 75  
mark-pos . . . . . 67, 68, 124  
mark-ref . . . . . 67, 70, 72, 74  
mini-env . . . . . 27, 34, 47, 54, 55, 70, 95, 105, 106, 108, 113  
mini-right\* . . . . . 27, 30, 47, 70, 106, 108  
mini-right . . . . . 27, 30, 47, 54, 70, 106, 108  
mini-sep . . . . . 27, 47, 70, 95  
no-store . . . . . 29, 62–65, 70, 76, 85, 86



noitemsep .....	42
nosep .....	42
parindent .....	92
parsep .....	42, 92, 111
partopsep .....	42
ref .....	26, 30, 38–40, 125
resume* .....	26, 58, 59, 62, 63, 70, 96, 108, 120
resume .....	26, 33, 58–63, 70, 96, 108, 120
rightmargin .....	43, 103
save-ans .....	28, 33, 58–63, 65, 66, 68–70, 76–78, 81–83, 89, 97, 99, 114, 115, 117, 118, 120, 125
save-key .....	28, 58, 69, 93, 108
save-pos .....	70
save-ref .....	29, 35, 67, 70, 72–74, 82, 84, 89, 115
save-sep .....	67, 70, 115
series .....	26, 58–62, 70, 93, 96, 108, 120
show-ans .....	67, 68, 70, 72, 73, 75, 89, 115
show-length .....	31, 45, 125
show-pos .....	28, 67, 68, 72, 73, 75, 84, 89, 115
start* .....	27, 41, 58
start .....	27, 30, 41, 58
store-key .....	68
topsep .....	42
widest .....	26, 30, 41
wrap-ans .....	35, 67, 70, 72, 74
wrap-label* .....	27, 37, 86, 87, 90, 110, 111, 114
wrap-label .....	27, 37, 85–87, 90, 110, 111, 114
wrap-opt .....	67, 70
keys commands:	
\keys_define:nn .....	489, 511, 543, 559, 606, 677, 751, 777, 821, 857, 880, 953, 962, 1041, 1058, 1483, 1594, 1837, 1898, 2057, 2099, 2135, 2140, 2454, 2605, 2641, 3079, 3095, 3115, 3135, 4592, 4691, 4807, 4815
\keys_if_exist_p:nn .....	4803, 4804
\l_keys_key_str .....	75, 79, 2472, 2627, 3103, 3121, 3143, 4850, 4958
\keys_precompile:nnN ..	118, 191, 191, 4594, 4598, 4602, 4606, 4610, 4614, 4833
\keys_set:nn ..	503, 837, 848, 1064, 1488, 1493, 1731, 1736, 1823, 1831, 2492, 3371, 3376, 3569, 4115, 4366, 4646, 4653, 4695, 4700, 4701, 4702, 4703, 4706, 4711, 4712, 4713, 4714, 4715, 4716, 4717, 4749, 4859
\keys_set_known:nn .....	2759
keyval commands:	
\keyval_parse:NNn .....	1608, 2165, 4667

L

label .....	557, 604, 677
Labels provide by enumext:	
\Alph* .....	36
\Roman* .....	36
\alph* .....	36
\arabic* .....	30, 36
\roman* .....	36
\labelsep .....	101
\labelsep .....	3727, 3730
labelsep .....	487
\labelwidth .....	36, 101
\labelwidth .....	3727, 3728
labelwidth .....	487
\lastkern .....	1083
\lastnodetype .....	1072
\lastskip .....	1077
\leftmargin .....	91
\leftmargin .....	90, 3727

legacy commands:	
\legacy_if:nTF .....	4227, 4230, 4475, 4478
\legacy_if_gset_false:n .....	375
\legacy_if_set_false:n .....	4229, 4477
\legacy_if_set_true:n ..	4189, 4214, 4221, 4234, 4413, 4441, 4482
\linewidth .....	95
\linewidth .....	3340, 3428, 3534, 3579, 3754, 3814, 3845, 3967, 4022
\list .....	359
list-indent .....	855
list-offset .....	855
\listparindent .....	3729
listparindent .....	855
\lrbox .....	4245, 4488

M

\makebox .....	103
\makebox ..	2270, 2272, 3046, 4259, 4267, 4271, 4501, 4505
\makelabel .....	85, 87, 90, 102
\makelabel .....	85, 89, 3063, 3197
\makesavenoteenv .....	408
mark-ans .....	2055
mark-pos .....	2055, 2097
mark-ref .....	2055
mini-env .....	1039
mini-sep .....	1039
\minipage .....	365
\miniright .....	10, 54, 1396, 1448, 1476, 3495, 3631
mode commands:	
\mode_if_math:TF .....	2535, 2589
\mode_if_vertical:TF ..	1135, 1164, 1185, 1269, 1288, 1317
\mode_leave_vertical: ..	835, 846, 908, 922, 934, 946, 2268, 3044, 4257
msg commands:	
\msg_error:nn ..	1450, 1478, 2496, 2529, 2533, 2587, 2695, 3560, 3564, 3653, 3713, 3744, 4100, 4347, 4359, 4718, 4777
\msg_error:nnn ..	582, 629, 646, 699, 1400, 1407, 1412, 1446, 1474, 1743, 1747, 1862, 2478, 2537, 2555, 2567, 2575, 2579, 2583, 2591, 2633, 3109, 3127, 3149, 4104, 4352, 4580, 4589, 4660, 4765, 4796, 4805, 4842, 4863
\msg_error:nnnn ..	2481, 2509, 2513, 2517, 2521, 2636, 3112, 3130, 3152, 3551, 3649, 3657, 4641, 4845
\msg_error:nnnnn .....	530, 550, 2078
\msg_fatal:nn .....	3359
\msg_fatal:nnn .....	441
\msg_info:nnn .....	13, 16, 21, 24, 390, 404
\msg_line_context: ..	4923, 4928, 4933, 4962, 4967, 4972, 4987, 5002, 5006, 5010, 5014, 5018, 5022, 5029, 5036, 5042, 5056, 5060, 5065, 5069, 5073, 5077, 5082, 5086, 5090, 5094, 5099, 5134, 5138, 5143, 5148, 5152, 5157, 5233, 5237, 5242, 5247, 5252, 5256, 5260, 5264, 5268, 5272, 5276, 5280, 5284
\msg_log:nnn .....	1882, 1887, 1892
\msg_log:nnnnn .....	352, 2025, 2030, 2035
\msg_log:nnnnnn .....	344
\msg_new:nnn ..	4890, 4894, 4898, 4902, 4907, 4920, 4925, 4930, 4935, 4944, 4952, 4956, 4960, 4965, 4970, 4985, 5000, 5004, 5008, 5012, 5016, 5020, 5024, 5033, 5039, 5045, 5049, 5053, 5058, 5063, 5067, 5071, 5075, 5080, 5084, 5088, 5092, 5097, 5132, 5136, 5141, 5146, 5150, 5155, 5231, 5235, 5240, 5245, 5250, 5254, 5258, 5262, 5266, 5270, 5274, 5278, 5282

<code>\msg_new:nnnn</code> . . . . .	4911, 5102, 5111, 5120, 5126, 5159, 5169, 5179, 5189, 5199, 5209, 5219, 5225
<code>\msg_term:nnnn</code> . . . . .	1846, 1851, 3265, 3275, 3306, 3311
<code>\msg_term:nnnnn</code> . . . . .	2006
<code>\msg_warning:nn</code> . . . . .	3494, 3630
<code>\msg_warning:nnnn</code> 2043, 2049, 3213, 3218, 3868, 3881, 3917, 3930	
<code>\msg_warning:nnnnn</code> . . . . .	2001, 2011
<code>\multicolsep</code> . . . . .	95
<code>\multicolsep</code> . . . . .	1204, 1311, 3463, 3606
<b>N</b>	
<code>\NeedsTeXFormat</code> . . . . .	3
<code>\newcounter</code> . . . . .	444
<code>\NewDocumentCommand</code> 1396, 2484, 3645, 4564, 4619, 4725, 4774, 4852	
<code>\NewDocumentEnvironment</code> . . . . .	3316, 3511, 3686, 4070, 4320
<code>\newenvsc</code> . . . . .	2598
<code>\newlabel</code> . . . . .	35
<code>\newlabel</code> . . . . .	426
<code>no-store</code> . . . . .	1896
<code>\noindent</code> . . . . .	3976, 4031, 4083, 4300, 4332, 4543
<code>\nointerlineskip</code> 1271, 1274, 1319, 1322, 1438, 1466, 3976, 4031	
<code>noitemsep</code> . . . . .	775
<code>\nopagebreak</code> 1147, 1176, 1271, 1274, 1319, 1322, 1387, 1393	
<code>\normalfont</code> . . . . .	2443, 2966, 2979, 4461
<code>nosep</code> . . . . .	775
<b>P</b>	
Packages:	
<code>caption</code> . . . . .	106
<code>enumext</code> . . . . .	25, 35, 38, 62, 90, 99, 123, 124
<code>enumitem</code> . . . . .	35, 36
<code>expl3</code> . . . . .	102
<code>footnotehyper</code> . . . . .	34
<code>hyperref</code> . . . . .	29, 30, 34, 35, 74, 84, 110, 123
<code>lua-visual-debug</code> . . . . .	50
<code>multicol</code> . . . . .	25, 123
<code>scontents</code> . . . . .	25, 77, 78
<code>shortlst</code> . . . . .	102
<code>\par</code> . . . . .	1147, 1176, 1274, 1322, 1387, 1393, 1431, 1438, 1466, 2420, 3480, 3484, 3620, 3636, 3766, 3994, 4008, 4049, 4063, 4300, 4314, 4543, 4559
<code>\parbox</code> . . . . .	2062
<code>\parindent</code> . . . . .	4277, 4511
<code>\parsep</code> . . . . .	48, 100, 101
<code>\parsep</code> . . . . .	3297, 3726, 3733, 3738
<code>parsep</code> . . . . .	775
<code>\parskip</code> . . . . .	4278, 4512
<code>\partopsep</code> . . . . .	101
<code>\partopsep</code> . . . . .	3298, 3731
<code>partopsep</code> . . . . .	775
peek commands:	
<code>\peek_meaning:N</code> 4166, 4180, 4197, 4208, 4390, 4404, 4421	
<code>\peek_meaning_remove:N</code> . . . . .	4173, 4397
<code>\peek_remove_spaces:n</code> . . . . .	3185
<code>\phantomsection</code> . . . . .	35
<code>\phantomsection</code> . . . . .	415
prg commands:	
<code>\prg_do_nothing:</code> . . . . .	419
<code>\prg_new_protected_conditional:Npnn</code> . . . . .	209
<code>\prg_replicate:nn</code> . . . . .	226
<code>\prg_return_false:</code> . . . . .	213
<code>\prg_return_true:</code> . . . . .	212
<code>\printkeyans</code> . . . . .	16, 118, 4619
prop commands:	
<code>\prop_const_from_keyval:Nn</code> . . . . .	4766
<code>\prop_count:N</code> 346, 2194, 2345, 2446, 2856, 2969, 2982, 4464, 4868	
<code>\prop_get:NnNTF</code> . . . . .	4792
<code>\prop_gput_if_not_in:Nnn</code> . . . . .	2192
<code>\prop_if_exist:N</code> . . . . .	1880, 4584, 4861
<code>\prop_item:Nn</code> . . . . .	4586, 4885
<code>\prop_new:N</code> . . . . .	1883
<code>\ProvidesExplPackage</code> . . . . .	4
<b>R</b>	
<code>\raggedcolumns</code> . . . . .	3466, 3609
<code>\ref</code> . . . . .	72, 82
<code>ref</code> . . . . .	557, 604, 677
<code>\refstepcounter</code> . . . . .	4236, 4484
regex commands:	
<code>\regex_match:nnTF</code> . . . . .	211, 726, 728, 740, 742, 2691
<code>\regex_replace_once:nnN</code> . . . . .	219
<code>\renewcommand</code> . . . . .	592, 637, 654, 707
<code>\RenewDocumentCommand</code> 1448, 1476, 3052, 3063, 3181, 3197, 3742, 3776	
<code>\RequirePackage</code> . . . . .	17, 25
<code>resume</code> . . . . .	1592
<code>resume*</code> . . . . .	1592
<code>rightmargin</code> . . . . .	855
<code>\Roman</code> . . . . .	36, 41
<code>\Roman</code> . . . . .	463
<code>\roman</code> . . . . .	36, 41
<code>\roman</code> . . . . .	464, 575, 4609
<b>S</b>	
<code>\s</code> . . . . .	2692
<code>save-ans</code> . . . . .	1835
<code>save-key</code> . . . . .	2133
<code>save-ref</code> . . . . .	2055
<code>save-sep</code> . . . . .	2055
scan commands:	
<code>\scan_stop:</code> . . . . .	101, 3740, 4082, 4331, 4575, 4578
scontents internal commands:	
<code>\l_scontents_fname_out_tl</code> . . . . .	2651
<code>\__scontents_parse_environment_keys:n</code> . . . . .	2657
<code>\__scontents_rescan_tokens:n</code> . . . . .	2664
<code>\l_scontents_storing_bool</code> . . . . .	2649
<code>\l_scontents_writing_bool</code> . . . . .	2650
seq commands:	
<code>\seq_clear:N</code> . . . . .	4727, 4870
<code>\seq_const_from_clist:Nn</code> . . . . .	4720
<code>\seq_count:N</code> . . . . .	347, 3699, 4731
<code>\seq_gclear:N</code> . . . . .	3774, 3775
<code>\seq_gput_right:Nn</code> . . . . .	2201, 3787, 3788
<code>\seq_if_empty:N</code> . . . . .	3793, 4634, 4745
<code>\seq_if_exist:N</code> . . . . .	1885, 4632
<code>\seq_if_in:NnTF</code> . . . . .	4639
<code>\seq_item:Nn</code> . . . . .	2689, 3763
<code>\seq_map_function:NN</code> . . . . .	4736
<code>\seq_map_inline:Nn</code> . . . . .	4647, 4654, 4746, 4747
<code>\seq_map_pairwise_function:NNN</code> . . . . .	3795
<code>\seq_new:N</code> . . . . .	122, 123, 125, 139, 163, 164, 1888
<code>\seq_pop_left:NN</code> . . . . .	4735
<code>\seq_put_right:Nn</code> . . . . .	3659, 4743, 4759, 4880
<code>\seq_set_from_clist:Nn</code> . . . . .	4728
<code>\seq_set_map_e:NNn</code> . . . . .	4737
<code>\seq_show:N</code> . . . . .	4636

\seq_use:Nn	191, 192, 4876
series	1592
\setcounter	737, 741, 743, 3256, 3300, 3704
\setenumext	6, 119, 4725
\setenumextmeta	6, 121, 4766
show-ans	2055, 2097
show-length	951
show-pos	2097
skip commands:	
\skip_add:Nn	1115, 1121, 1127, 1137, 1141, 1166, 1170, 1187, 1228, 1229, 1240, 1241, 1248, 1257, 1290, 1301, 3726
\skip_gset:Nn	1335, 1339, 1343
\skip_gzero_new:N	1330, 1331
\skip_horizontal:N	923, 935, 947, 4260, 4274, 4508
\skip_horizontal:n	909, 2269, 2277, 3045, 3047, 4258, 4517
\skip_if_eq:nnTF	1113, 1119, 1125, 1196, 1216, 1246, 1255, 1297, 1299, 1332, 1354, 1500, 1514, 1528, 1539, 1550, 1561, 1572, 1583
\skip_new:N	81, 82, 83, 87, 88, 89, 90, 91, 143, 183
\skip_set:Nn	1098, 1102, 1152, 1156, 1181, 1205, 1208, 1219, 1220, 1234, 1284, 1334, 1338, 1356, 1360, 1364, 1370, 1374, 1378, 3720, 3734
\skip_set_eq:NN	1192, 1193, 1194, 1195, 1204, 1295, 1311, 3254, 3296, 3297, 4277, 4278, 4511, 4512
\skip_sub:Nn	1226, 1227, 1238, 1239, 1251, 1260, 1304
\skip_use:N	1100, 1104, 1139, 1143, 1148, 1168, 1172, 1183, 1189, 1501, 1505, 1508, 1515, 1519, 1522, 3480, 3484
\skip_vertical:N	376, 379, 1429, 1460
\skip_zero:N	1203, 1275, 1281, 1282, 1283, 1310, 1323, 3298, 3463, 3606, 3731, 3732
\skip_zero_new:N	1329, 1351, 1352, 1353
\l_tmpa_skip	1234, 1240, 1241
\c_zero_skip	376, 379, 1113, 1119, 1125, 1196, 1246, 1255, 1297, 1299, 1332, 1354, 1501, 1515, 1528, 1539, 1550, 1561, 1572, 1583
\small	4597, 4601, 4605, 4609, 4613, 4617
\star	3083
start	749
start*	749
\stepcounter	3666, 3780
str commands:	
\c_backslash_str	2537, 4923, 4928, 4933, 4938, 4940, 4942, 4947, 4949, 5047, 5051, 5055, 5065, 5069, 5077, 5078, 5082, 5094, 5095, 5099, 5100, 5121, 5123, 5127, 5129, 5157, 5220, 5222, 5226, 5228, 5237, 5238, 5242, 5247, 5248, 5252, 5256, 5260
\c_colon_str	2344, 2855, 4575
\c_left_brace_str	5028, 5035, 5041
\c_right_brace_str	5028, 5035, 5041
\str_case:nn	231, 290
\str_case:nnTF	1615, 1623, 2172, 2180, 4674, 4683
\str_clear:N	3368, 4114
\str_count:n	226
\str_if_empty:NTF	1632, 1673, 1700
\str_if_eq:nnTF	3257, 3302, 4776
\str_if_in:nnTF	4571
\str_new:N	129, 178
\str_set:Nn	546, 547, 548, 2075, 2076, 2102, 2103
\string	408
\strutbox	1205, 1208, 1219, 1220, 1228, 1229, 1240, 1241, 1248, 1257, 1301, 1325, 1334, 1335, 1338, 1345, 1358, 1366, 1372, 1380, 3736

T	
T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@auxout	424
\@currentvir	231, 290
\protected@write	424
tex commands:	
\tex_newlinechar:D	2663
text commands:	
\text_expand:n	4567
\textasteriskcentered	2072, 2089
\the	1077, 1083
\thepage	430
tl commands:	
\c_space_tl	2937, 4972, 4987, 5010, 5014, 5201, 5202, 5211, 5212, 5272, 5276
\tl_clear:N	516, 522, 2053, 2119, 2129, 2150, 2158, 2364, 2683, 2684, 2798, 2872, 4427
\tl_clear_new:N	473
\tl_const:Nn	50, 457
\tl_gclear:N	338, 339, 340, 1653, 1658, 2773, 3074, 4012, 4067, 4261
\tl_gclear_new:N	1640
\tl_gput_right:Nn	458
\tl_greplace_all:Nnn	479
\tl_gset:Nn	266, 267, 281, 282, 1641, 1654, 1659, 1878, 2687, 3021, 4203
\tl_gset_eq:NN	475, 3017, 4254
\tl_if_blank:nTF	2476, 2494, 2631, 3107, 3125, 3147, 4252, 4840
\tl_if_empty:NTF	580, 599, 627, 644, 664, 671, 697, 714, 1666, 1671, 1693, 1698, 1756, 1820, 1828, 1857, 1917, 2208, 2239, 2384, 2728, 2750, 2780, 2809, 2882, 2931, 3042, 4430, 4757
\tl_if_empty:nTF	1721
\tl_if_exist:NTF	1726
\tl_if_novalue:nTF	2490, 2806, 2880, 2916, 2996, 3015, 3023, 3157, 3366, 3697, 3778, 4112, 4364, 4428
\tl_map_inline:Nn	217, 476
\tl_new:N	42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 103, 104, 105, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 124, 126, 127, 128, 130, 133, 134, 146, 154, 155, 156, 159, 177
\tl_put_left::Ne	2717
\tl_put_left:Nn	2216, 2247, 2369, 2711, 2724, 2730, 2740, 2948, 2988, 3997, 4052, 4446, 4449
\tl_put_right:Nn	474, 590, 635, 652, 705, 2220, 2251, 2298, 2308, 2321, 2336, 2342, 2347, 2371, 2376, 2383, 2386, 2396, 2401, 2404, 2410, 2801, 2804, 2811, 2813, 2840, 2845, 2850, 2853, 2862, 2875, 2878, 2884, 2889, 2899, 4432, 4433
\tl_remove_all:Nn	4756
\tl_remove_once:Nn	2286, 2825
\tl_replace_all:Nnn	478, 4791
\tl_reverse:N	2285, 2287, 2824, 2826
\tl_set:Nn	58, 235, 245, 294, 295, 302, 303, 310, 311, 443, 517, 521, 526, 527, 579, 624, 696, 906, 920, 932, 944, 1755, 1856, 2120, 2130, 2151, 2159, 2440, 2651, 2918, 2963, 2976, 4435, 4458, 4754, 4790, 4860
\tl_set_eq:NN	484, 585, 588, 632, 634, 649, 651, 702, 704, 2284, 2823, 2836, 3169, 3173, 3678, 3680
\tl_to_str:n	1726, 1732, 1737, 4567
\tl_trim_spaces:n	474, 4743, 4754, 4760, 4776
\tl_use:N	480, 483, 601, 666, 673, 716, 977, 981, 985, 989, 993, 997, 1001, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 2274, 2291, 2299, 2310, 2323,

2328, 2339, 3004, 3010, 3038, 3065, 3066, 3073, 3160, 3164, 3172, 3199, 3200, 3206, 3323, 3517, 3683, 4004, 4059, 4264, 4275, 4279, 4498, 4509, 4515, 4520, 4622, 4623, 4624, 4625, 4626, 4644, 4739, 4858	\use_none:nn . . . . . 418, 4797
token commands:	\usecounter . . . . . 3255, 3299
\token_to_str:N . . . . . 426	<b>V</b>
topsep . . . . . 775	\value . . . . . 1669, 1675, 1682, 1688, 1696, 1702, 1709, 1715
\topskip . . . . . 1203, 1310	vbox commands:
\typeout . . . . . 394, 397, 407, 408, 1076, 1077, 1082, 1083, 1088, 1218, 1225, 1233	\vbox_set_top:Nn . . . . . 4002, 4057
<b>U</b>	\vspace . . . . . 836, 847, 1505, 1508, 1519, 1522, 1532, 1534, 1543, 1545, 1554, 1556, 1565, 1567, 1576, 1578, 1587, 1589, 3694, 3705, 4315, 4560
\u . . . . . 220, 2692	<b>W</b>
\unkern . . . . . 1084	widest . . . . . 749
unknown . . . . . 3093, 3115, 3133	wrap-ans . . . . . 2055
\unpenalty . . . . . 1089	wrap-label . . . . . 487
\unskip . . . . . 1078, 3634	wrap-label* . . . . . 487
use commands:	wrap-opt . . . . . 2055
\use:N . . . . . 227, 3070, 3325	<b>Z</b>
\use:n . . . . . 1606, 2163, 4573, 4665	\z . . . . . 2692