

# 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  $\text{\TeX-SX}$  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 scontents[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  >> TDS:tex/latex/enumext/
enumext.pdf  >> TDS:doc/latex/enumext/
README.md   >> TDS:doc/latex/enumext/
enumext.dtx  >> TDS:source/latex/enumext/
```

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

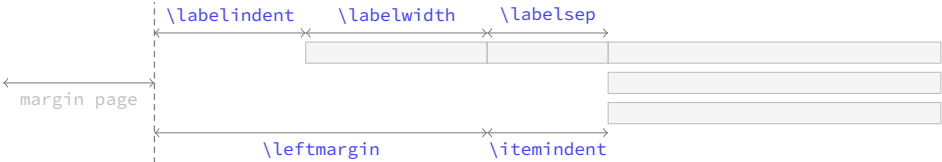


Figure 1: Representation of horizontal lengths in `enumitem`.

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

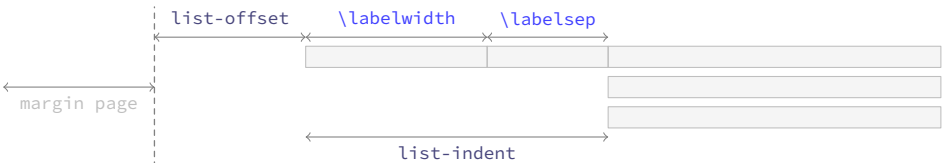


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

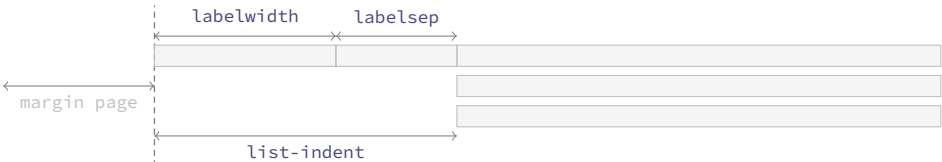


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

### 1.3 User interface

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

#### 1.3.1 Internal counters

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

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

#### 1.3.2 Public dimension

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

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

#### 1.3.3 Support for multicol

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

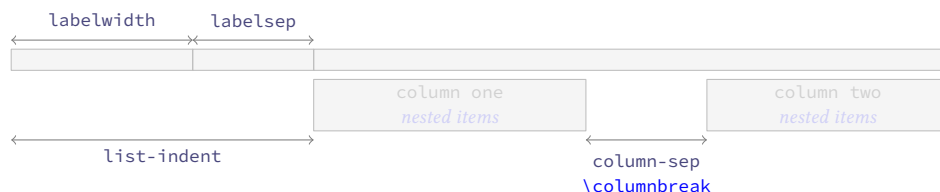


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

#### 1.3.4 Support for minipage

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



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

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

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

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

- 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 `LaTeX`, `\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 “discordable”.

`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 discordable”.

`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  $\{\langle text symbol \rangle\}$  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]{\langle drawing \rangle}{\langle 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.

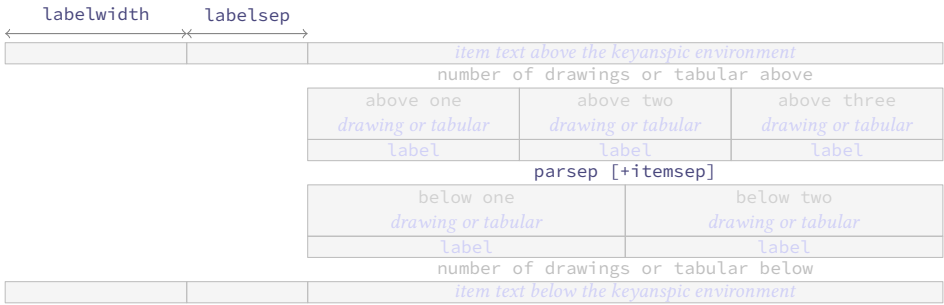


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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* `{\langle store name \rangle}` set by `save-ans` key.

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

Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
\item Question with images.
\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 <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 <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
- ©2024 by Pablo González L
- 19 / 144





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** 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 `{⟨store name⟩}` 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 `{⟨store name⟩}` 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 `⟨body⟩` and the `⟨keys⟩` 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).

`\seq_use:NV`

```

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.)

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

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

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 \l__enumext_counter_i_tl { enumXi }
enumXv    448 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXvi   449 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii  450 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii 451 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
          452 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
          453 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
          454 \__enumext_define_counters:Nn \l__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. 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.

`\__enumext_label_style:cvn`

```

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.

```

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:cnv { \__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:cnv { \__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.

```

topsep 775 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
partopsep 776 {
parsep 777   \keys_define:nn { enumext / #1 }
noitemsep 778   {
nosep 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                 \ignorespaces
912             }
913         }
914     }
915 \cs_set_protected:Nn l__enumext_keyans_fake_item:
916 {
917     \dim_compare:nNnT
918     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
919     {
920         \tl_set:Nc l__enumext_fake_item_indent_v_tl
921         {

```

```

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 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  $\text{\TeX}$  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:Nn \__enumext_multi_set_vskip:
1071 {
1072   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1073   {
1074     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1075   }
1076   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1077   {
1078     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1079   }
1080   \__enumext_add_pre_parsep:
1081 }

```

(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*.

```

1082 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1083 {
1084   \int_case:nn { \l__enumext_level_int }
1085   {
1086     { 2 }{
1087       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1088       {
1089         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1090       }
1091     }
1092     { 3 }{
1093       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1094       {
1095         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1096       }
1097     }
1098     { 4 }{
1099       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1100       {
1101         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1102       }
1103     }
1104   }
1105 }

```

(End of definition for `\__enumext_add_pre_parsep:`)

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

```

1106 \cs_new_protected:Nn \__enumext_multi_addvspace:
1107 {
1108   \__enumext_multi_set_vskip:
1109   \mode_if_vertical:T
1110   {
1111     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1112     {
1113       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1114     }
1115     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1116     {
1117       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1118     }
1119   }
1120   \par\nopagebreak
1121   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1122 }

```

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

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

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

`\__enumext_keyans_multi_addvspace:`

```

1123 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1124 {
1125   \skip_set:Nn \l__enumext_multicols_above_v_skip
1126   {
1127     \l__enumext_topsep_v_skip
1128   }
1129   \skip_set:Nn \l__enumext_multicols_below_v_skip
1130   {
1131     \l__enumext_topsep_v_skip
1132   }
1133 }
1134 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1135 {
1136   \__enumext_keyans_multi_set_vskip:
1137   \mode_if_vertical:T
1138   {
1139     \skip_add:Nn \l__enumext_multicols_above_v_skip
1140     {
1141       \skip_use:N \l__enumext_partopsep_v_skip
1142     }
1143     \skip_add:Nn \l__enumext_multicols_below_v_skip
1144     {
1145       \skip_use:N \l__enumext_partopsep_v_skip
1146     }
1147   }
1148   \par\nopagebreak
1149   \addvspace{ \l__enumext_multicols_above_v_skip }
1150 }

```

(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.

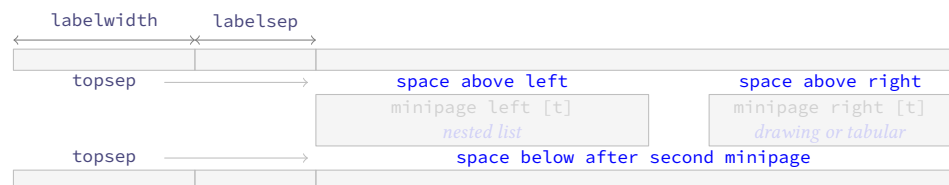


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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`<sup>[14]</sup> 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`.

```

1151 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1152 {
1153   \skip_set:Nn \l__enumext_minipage_right_skip
1154   {
1155     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1156   }
1157   \mode_if_vertical:T
1158   {
1159     \skip_add:Nn \l__enumext_minipage_right_skip
1160     {
1161       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1162     }
1163   }
1164   \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1165   \skip_if_eq:nnF { \l__enumext_minipage_after_skip } { \c_zero_skip }
1166   {
1167     \__enumext_previous_level_skip:
1168   }

```

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`

```

1169   \int_compare:nNnT
1170   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1171   {
1172     \skip_zero:N \topskip
1173     \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1174   }
1175 }
1176 \cs_new_protected:Nn \__enumext_previous_level_skip:
1177 {
1178   \int_case:nn { \l__enumext_level_int }
1179   {
1180     { 2 } {
1181       \skip_if_eq:nnTF { \l__enumext_itemsep_i_skip } { \c_zero_skip }
1182       {
1183         \skip_add:Nn \l__enumext_minipage_after_skip { 0.15\box_ht:N \strutbox }
1184       }
1185       {
1186         \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1187       }
1188     }
1189     { 3 } {
1190       \skip_if_eq:nnTF { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1191       {
1192         \skip_add:Nn \l__enumext_minipage_after_skip { 0.15\box_ht:N \strutbox }
1193       }
1194       {
1195         \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1196       }
1197     }

```

```

1198         { 4 }{
1199             \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1200             {
1201                 \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1202             }
1203             {
1204                 \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1205             }
1206         }
1207     }
1208 }

```

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

```

1209 \cs_new_protected:Nn \__enumext_minipage_add_space:
1210 {
1211     \__enumext_minipage_set_skip:
1212     \mode_if_vertical:TF
1213     {
1214         \nopagebreak\nointerlineskip
1215     }
1216     {
1217         \par\nopagebreak\nointerlineskip
1218         \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1219     }
1220     \addvspace{ 0.245\box_ht:N \strutbox }
1221 }

```

(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*.

```

1222 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1223 {
1224     \skip_zero:N \l__enumext_minipage_after_skip
1225     \skip_zero:N \l__enumext_minipage_left_skip
1226     \skip_zero:N \l__enumext_minipage_right_skip
1227     \skip_set:Nn \l__enumext_minipage_right_skip
1228     {
1229         \l__enumext_topsep_v_skip
1230     }
1231     \mode_if_vertical:T
1232     {
1233         \skip_add:Nn \l__enumext_minipage_right_skip
1234         {
1235             \l__enumext_partopsep_v_skip
1236         }
1237     }
1238     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1239     %% prev level
1240     \skip_if_eq:nnF { \l__enumext_minipage_after_skip } { \c_zero_skip }
1241     {
1242         \skip_if_eq:nnTF { \l__enumext_itemsep_i_skip } { \c_zero_skip }
1243         {
1244             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1245         }
1246         {
1247             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1248         }
1249     }
1250     %% columns
1251     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1252     {
1253         \skip_zero:N \topskip
1254         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1255     }

```



```
1256 }
```

(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 T<sub>E</sub>X 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`.

```
1257 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1258 {
1259   \__enumext_keyans_minipage_set_skip:
1260   \mode_if_vertical:TF
1261   {
1262     \nopagebreak\nointerlineskip
1263   }
1264   {
1265     \par\nopagebreak\nointerlineskip
1266     \skip_zero:N \l__enumext_partopsep_v_skip
1267   }
1268   \addvspace{ 0.245\box_ht:N \strutbox }
1269 }
```

(End of definition for `\__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*`.

```
1270 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1271 {
1272   \skip_zero_new:N \l__enumext_minipage_left_skip
1273   \skip_gzero_new:N \g__enumext_minipage_right_skip
1274   \skip_gzero_new:N \g__enumext_minipage_after_skip
1275   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1276   {
1277     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1278     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1279   }
1280   {
1281     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1282     \skip_gset:Nn \g__enumext_minipage_right_skip
1283     {
1284       \l__enumext_topsep_vii_skip
1285     }
1286     \skip_gset:Nn \g__enumext_minipage_after_skip
1287     {
1288       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1289     }
1290   }
1291 }
1292 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1293 {
1294   \skip_zero_new:N \l__enumext_minipage_after_skip
1295   \skip_zero_new:N \l__enumext_minipage_left_skip
1296   \skip_zero_new:N \l__enumext_minipage_right_skip
1297   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1298   {
1299     \skip_set:Nn \l__enumext_minipage_left_skip
1300     {
1301       0.5\box_dp:N \strutbox
1302     }
1303     \skip_set:Nn \l__enumext_minipage_right_skip
1304     {
1305       \l__enumext_partopsep_viii_skip
1306     }
1307     \skip_set:Nn \l__enumext_minipage_after_skip
1308     {
1309       1.6\box_dp:N \strutbox
1310     }
1311 }
```

```

1311     }
1312     {
1313         \skip_set:Nn \__enumext_minipage_left_skip
1314         {
1315             0.5875\box_dp:N \strutbox
1316         }
1317         \skip_set:Nn \__enumext_minipage_right_skip
1318         {
1319             \l__enumext_topsep_viii_skip
1320         }
1321         \skip_set:Nn \__enumext_minipage_after_skip
1322         {
1323             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1324         }
1325     }
1326 }

```

(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  $\TeX$  is in  $\langle horizontal mode \rangle$  or  $\langle vertical mode \rangle$ , since `\partopsep` is equal to `0pt` in both environments.

```

1327 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1328 {
1329     \__enumext_mini_set_vskip_vii:
1330     \par\nopagebreak
1331     \addvspace { \l__enumext_minipage_left_skip }
1332 }
1333 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1334 {
1335     \__enumext_mini_set_vskip_viii:
1336     \par\nopagebreak
1337     \addvspace { \l__enumext_minipage_left_skip }
1338 }

```

(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  $\TeX$  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.

```

1339 \NewDocumentCommand \miniright { s }
1340 {
1341     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1342     {
1343         \msg_error:nnn { enumext } { wrong-miniright-place }
1344     }
1345     % outside
1346     \bool_lazy_and:nnT
1347     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1348     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1349     {
1350         \msg_error:nnn { enumext } { wrong-miniright-place }
1351     }
1352     % starred env
1353     \bool_if:NT \l__enumext_starred_bool
1354     {
1355         \msg_error:nnn { enumext } { wrong-miniright-starred }
1356     }
1357     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1358     {
1359         \__enumext_keyans_mini_right_cmd:n {#1}

```

```

1360     }
1361     { \__enumext_mini_right_cmd:n {#1} }
1362 }

```

(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 `\__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.

```

1363 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1364 {
1365     \dim_compare:nNnTF
1366     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1367     {
1368         \__enumext_multicols_stop:
1369         \int_compare:nNnT
1370         { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } = { 1 }
1371         {
1372             \skip_vertical:N \__enumext_minipage_after_skip
1373         }
1374         \end{__enumext_mini_env*}
1375         \hfill
1376         \begin{__enumext_mini_env*}
1377         { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1378         % Add vertical space above
1379         \par\nointerlineskip
1380         \addvspace { \__enumext_minipage_right_skip }
1381         \bool_if:nF {#1}
1382         {
1383             \centering
1384         }
1385         \int_gzero:N \g__enumext_minipage_stat_int
1386     }
1387     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1388 % paranoia
1389 \RenewDocumentCommand \miniright { s }
1390 {
1391     \msg_error:nn { enumext } { many-miniright-used }
1392 }
1393 }

```

(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.

```

1394 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1395 {
1396     \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1397     {
1398         \__enumext_keyans_multicols_stop:
1399         \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1400         {
1401             \skip_vertical:N \__enumext_minipage_after_skip
1402         }
1403         \end{__enumext_mini_env*}
1404         \hfill
1405         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1406         % Add vertical space above
1407         \par\nointerlineskip
1408         \addvspace { \__enumext_minipage_right_skip }
1409         \bool_if:nF {#1}
1410         {
1411             \centering
1412         }
1413         \int_gzero:N \g__enumext_minipage_stat_int
1414     }
1415     { \msg_error:nnn { enumext } { wrong-miniright-use } }

```

```

1416 % paranoia
1417 \RenewDocumentCommand \miniright { s }
1418 {
1419     \msg_error:nn { enumext } { many-miniright-used }
1420 }
1421 }

```

(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.

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

```

above* 1422 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below   1423 {
below*  1424     \keys_define:nn { enumext / #1 }
        1425     {
        1426         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
        1427         above .value_required:n = true,
        1428         above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
        1429                     \keys_set:nn { enumext / #1 } { above = {##1} },
        1430         above* .value_required:n = true,
        1431         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
        1432         below .value_required:n = true,
        1433         below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
        1434                     \keys_set:nn { enumext / #1 } { below = {##1} },
        1435         below* .value_required:n = true,
        1436     }
        1437 }
        1438 \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.

```

1439 \cs_new_protected:Nn \__enumext_vspace_above:
1440 {
1441     \skip_if_eq:nnF
1442     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1443     {
1444         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1445         {
1446             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1447         }
1448         {
1449             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1450         }
1451     }
1452 }

```

(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.

```

1453 \cs_new_protected:Nn \__enumext_vspace_below:
1454 {
1455     \skip_if_eq:nnF
1456     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1457     {
1458         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1459         {
1460             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1461         }
1462         {
1463             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }

```

```

1464     }
1465   }
1466 }

```

(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.

```

1467 \cs_new_protected:Nn \__enumext_vspace_above_v:
1468 {
1469   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1470   {
1471     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1472     {
1473       \vspace*{ \__enumext_vspace_above_v_skip }
1474     }
1475     { \vspace { \__enumext_vspace_above_v_skip } }
1476   }
1477 }

```

(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.

```

1478 \cs_new_protected:Nn \__enumext_vspace_below_v:
1479 {
1480   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1481   {
1482     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1483     {
1484       \vspace*{ \__enumext_vspace_below_v_skip }
1485     }
1486     { \vspace { \__enumext_vspace_below_v_skip } }
1487   }
1488 }

```

(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:`

```

1489 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1490 {
1491   \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1492   {
1493     \bool_if:NTF \__enumext_vspace_a_star_vii_bool
1494     {
1495       \vspace*{ \__enumext_vspace_above_vii_skip }
1496     }
1497     { \vspace { \__enumext_vspace_above_vii_skip } }
1498   }
1499 }
1500 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1501 {
1502   \skip_if_eq:nnF { \__enumext_vspace_above_viii_skip } { \c_zero_skip }
1503   {
1504     \bool_if:NTF \__enumext_vspace_a_star_viii_bool
1505     {
1506       \vspace*{ \__enumext_vspace_above_viii_skip }
1507     }
1508     { \vspace { \__enumext_vspace_above_viii_skip } }
1509   }
1510 }

```

(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:`

```

1511 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1512 {

```

```

1513 \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1514 {
1515     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1516     {
1517         \vspace*{ \l__enumext_vspace_below_vii_skip }
1518     }
1519     { \vspace { \l__enumext_vspace_below_vii_skip } }
1520 }
1521 }
1522 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1523 {
1524     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1525     {
1526         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1527         {
1528             \vspace*{ \l__enumext_vspace_below_viii_skip }
1529         }
1530         { \vspace { \l__enumext_vspace_below_viii_skip } }
1531     }
1532 }

```

(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`

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1533 \cs_set_protected:Npn \__enumext_tmp:n #1
1534 {
1535     \keys_define:nn { enumext / #1 }
1536     {
1537         series .str_set:N = \l__enumext_series_str,
1538         series .value_required:n = true,
1539         resume .code:n = \__enumext_resume_series:n {##1},
1540         resume* .code:n = \__enumext_resume_starred:,
1541         resume* .value_forbidden:n = true,
1542     }
1543 }
1544 \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.

```

1545 \cs_new:Npn \__enumext_filter_series:n #1
1546 {
1547     \use:e
1548     {
1549         \keyval_parse:NNn
1550         \__enumext_filter_series_key:n
1551         \__enumext_filter_series_pair:nn {#1}
1552     }
1553 }

```

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.

```

1554 \cs_new:Npn \__enumext_filter_series_key:n #1
1555 {
1556     \str_case:nnF {#1}
1557     {
1558         { resume } {} { resume* } {} { base-fix } {}
1559     }
1560     { , { \exp_not:n {#1} } }
1561 }

```



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

```

1562 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1563 {
1564   \str_case:nnF {#1}
1565   {
1566     { series } {} { resume } {} { start } {}
1567     { start* } {} { save-ans } {} { save-key } {}
1568   }
1569   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1570 }

```

(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 `\__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 `\__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).

```

1571 \cs_new_protected:Npn \__enumext_parse_series:n #1
1572 {
1573   \str_if_empty:NTF \l__enumext_series_str
1574   {
1575     \bool_if:NF \l__enumext_resume_active_bool
1576     {
1577       \__enumext_resume_last:n {#1}
1578     }
1579   }
1580   {
1581     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1582     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1583     { \__enumext_filter_series:n {#1} }
1584     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1585     {
1586       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1587     }
1588   }
1589 }

```

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.

```

1590 \cs_new_protected:Npn \__enumext_resume_last:n #1
1591 {
1592   \bool_if:NT \l__enumext_standar_first_bool
1593   {
1594     \tl_gclear:N \g__enumext_standar_series_tl
1595     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1596   }
1597   \bool_if:NT \l__enumext_starred_first_bool
1598   {
1599     \tl_gclear:N \g__enumext_starred_series_tl
1600     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1601   }
1602 }

```

(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).

```

1603 \cs_new_protected:Nn \__enumext_resume_save_counter:
1604 {
1605   \bool_if:NT \g__enumext_standar_bool
1606   {
1607     \tl_if_empty:NF \l__enumext_series_str
1608     {
1609       \int_gset_eq:cN
1610       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1611     }
1612     \tl_if_empty:NTF \l__enumext_resume_name_tl
1613     {
1614       \str_if_empty:NT \l__enumext_series_str
1615       {
1616         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1617       }
1618     }
1619     {
1620       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1621       {
1622         \int_gset_eq:cN
1623         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1624       }
1625     }
1626     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1627     {
1628       \int_gset_eq:cN
1629       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1630     }
1631   }
1632   \bool_if:NT \g__enumext_starred_bool
1633   {
1634     \tl_if_empty:NF \l__enumext_series_str
1635     {
1636       \int_gset_eq:cN
1637       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1638     }
1639     \tl_if_empty:NTF \l__enumext_resume_name_tl
1640     {
1641       \str_if_empty:NT \l__enumext_series_str
1642       {
1643         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1644       }
1645     }
1646     {
1647       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1648       {
1649         \int_gset_eq:cN
1650         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1651       }
1652     }
1653     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1654     {
1655       \int_gset_eq:cN
1656       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1657     }
1658   }
1659 }

```

(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.

```

1660 \cs_new_protected:Npn \__enumext_resume_series:n #1
1661 {
1662   \tl_if_empty:nTF {#1}
1663   {
1664     \__enumext_resume_counter:n { }
1665   }
1666   {
1667     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1668     {
1669       \__enumext_resume_counter:n {#1}
1670       \bool_if:NT \g__enumext_standar_bool
1671       {
1672         \keys_set:nv { enumext / level-1 }
1673         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1674       }
1675       \bool_if:NT \g__enumext_starred_bool
1676       {
1677         \keys_set:nv { enumext / enumext* }
1678         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1679       }
1680     }
1681     {
1682       \bool_if:NT \g__enumext_standar_bool
1683       {
1684         \msg_error:nnn { enumext } { unknown-series } {#1}
1685       }
1686       \bool_if:NT \g__enumext_starred_bool
1687       {
1688         \msg_error:nnn { enumext } { unknown-series } {#1}
1689       }
1690     }
1691   }
1692 }

```

(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`.

```

1693 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1694 {
1695   \bool_set_true:N \l__enumext_resume_active_bool
1696   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1697   \tl_if_empty:NTF \l__enumext_resume_name_tl
1698   {
1699     \__enumext_resume_counter:
1700   }
1701   {
1702     \__enumext_resume_counter_series:
1703   }
1704   \__enumext_resume_counter_save_ans:
1705 }

```

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.

```

1706 \cs_new_protected:Nn \__enumext_resume_counter:
1707 {
1708   \bool_if:NT \g__enumext_standar_bool

```

```

1709     {
1710         \int_gincr:N \g__enumext_resume_int
1711         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1712     }
1713     \bool_if:NT \g__enumext_starred_bool
1714     {
1715         \int_gincr:N \g__enumext_resume_vii_int
1716         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1717     }
1718 }

```

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.

```

1719 \cs_new_protected:Nn \__enumext_resume_counter_series:
1720 {
1721     \bool_if:NT \g__enumext_standar_bool
1722     {
1723         \int_set:Nn \l__enumext_start_i_int
1724         {
1725             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1726         }
1727     }
1728     \bool_if:NT \g__enumext_starred_bool
1729     {
1730         \int_set:Nn \l__enumext_start_vii_int
1731         {
1732             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1733         }
1734     }
1735 }

```

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.

```

1736 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1737 {
1738     \bool_lazy_and:nnT
1739     { \bool_if_p:N \l__enumext_standar_first_bool }
1740     { \bool_if_p:N \l__enumext_store_active_bool }
1741     {
1742         \int_set:Nn \l__enumext_start_i_int
1743         {
1744             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1745         }
1746     }
1747     \bool_lazy_and:nnT
1748     { \bool_if_p:N \l__enumext_starred_first_bool }
1749     { \bool_if_p:N \l__enumext_store_active_bool }
1750     {
1751         \int_set:Nn \l__enumext_start_vii_int
1752         {
1753             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1754         }
1755     }
1756 }

```

(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.

```

1757 \cs_new_protected:Nn \__enumext_resume_starred:
1758 {
1759     \bool_if:NT \g__enumext_standar_bool
1760     {
1761         \tl_if_empty:NF \g__enumext_standar_series_tl
1762         {
1763             \__enumext_resume_counter:n { }

```

```

1764         \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1765     }
1766 }
1767 \bool_if:NT \g__enumext_starred_bool
1768 {
1769     \tl_if_empty:NF \g__enumext_starred_series_tl
1770     {
1771         \__enumext_resume_counter:n { }
1772         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1773     }
1774 }
1775 }

```

(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*`.

```

1776 \cs_set_protected:Npn \__enumext_tmp:n #1
1777 {
1778     \keys_define:nn { enumext / #1 }
1779     {
1780         save-ans .code:n = \__enumext_storing_set:n {##1},
1781         save-ans .value_required:n = true,
1782     }
1783 }
1784 \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:  
\\_\_enumext\_stop\_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:`.

```

1785 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1786 {
1787     \msg_term:nnVV { enumext } { save-ans-log }
1788     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1789 }
1790 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1791 {
1792     \msg_term:nnVV { enumext } { save-ans-log-hook }
1793     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1794 }

```

(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.

```

1795 \cs_new_protected:Npn \__enumext_storing_set:n #1
1796 {
1797     \tl_set:Nx \l__enumext_store_name_tl {#1}
1798     \tl_if_empty:NTF \l__enumext_store_name_tl
1799     {
1800         \bool_lazy_or:nnT
1801         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1802         {
1803             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1804         }
1805     }
1806     {
1807         \bool_lazy_or:nnT

```

```

1808         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1809         {
1810             __enumext_start_save_ans_msg:
1811             __enumext_storing_exec:
1812         }
1813     }
1814 }

```

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*`.

```

1815 \cs_new_protected:Nn \__enumext_storing_exec:
1816 {
1817     \bool_set_true:N \l__enumext_store_active_bool
1818     \bool_set_true:N \l__enumext_check_answers_bool
1819     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1820     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1821     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1822     {
1823         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1824         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1825     }
1826     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1827     {
1828         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1829         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1830     }
1831     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1832     {
1833         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1834         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1835     }
1836 }

```

(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  \cs_set_protected:Npn \__enumext_tmp:n #1
no-store   {
1837       \keys_define:nn { enumext / #1 }
1838       {
1839         {
1840           {
1841             check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1842             check-ans .initial:n = false,
1843             check-ans .value_required:n = true,
1844             no-store .code:n = {
1845               \bool_set_false:N \__enumext_check_answers_bool
1846               \bool_set_false:N \__enumext_check_ans_key_bool
1847             },
1848             no-store .value_forbidden:n = true,
1849           }
1850         }
1851       \clist_map_inline:nn
1852       {
1853         level-1, level-2, level-3, level-4, enumext*
1854       }
1855       { \__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 `\__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\__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.

```
1856 \cs_new_protected:Nn \__enumext_check_ans_active:
1857 {
1858   \tl_if_empty:NF \__enumext_store_name_tl
1859   {
1860     \bool_if:NT \__enumext_check_answers_bool
1861     {
1862       \__enumext_check_ans_level:
1863     }
1864   }
1865 }
```

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 `\__enumext_item_number_bool` to “*false*”.

```
1866 \cs_new_protected:Nn \__enumext_check_ans_level:
1867 {
1868   \int_case:nn { \__enumext_level_int }
1869   {
1870     { 1 }{
1871       \bool_lazy_all:NT
1872       {
1873         { \bool_if_p:N \g__enumext_starred_bool }
1874         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1875       }
1876       {
1877         \int_gdecr:N \g__enumext_item_number_int
1878         \bool_set_false:N \__enumext_item_number_bool
1879       }
1880     }
1881     { 2 }{
1882       \int_gdecr:N \g__enumext_item_number_int
1883       \bool_set_false:N \__enumext_item_number_bool
1884     }
1885     { 3 }{
1886       \int_gdecr:N \g__enumext_item_number_int
1887       \bool_set_false:N \__enumext_item_number_bool
1888     }
1889     { 4 }{
1890       \int_gdecr:N \g__enumext_item_number_int
1891       \bool_set_false:N \__enumext_item_number_bool
```



```

1892         }
1893     }

```

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.

```

1894     \int_case:nn { \l__enumext_level_h_int }
1895     {
1896         { 1 }{
1897             \bool_lazy_all:nT
1898             {
1899                 { \bool_if_p:N \g__enumext_standar_bool }
1900                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1901             }
1902             {
1903                 \int_gdecr:N \g__enumext_item_number_int
1904                 \bool_set_false:N \l__enumext_item_number_bool
1905             }
1906         }
1907     }
1908 }

```

(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.

```

1909 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1910 {
1911     \bool_lazy_and:nnT
1912     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1913     { \bool_if_p:N \g__enumext_standar_bool }
1914     {
1915         \bool_gset_true:N \g__enumext_check_ans_key_bool
1916     }
1917     \bool_lazy_and:nnT
1918     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1919     { \bool_if_p:N \g__enumext_starred_bool }
1920     {
1921         \bool_gset_true:N \g__enumext_check_ans_key_bool
1922     }
1923 }

```

(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:`.

```

1924 \cs_new_protected:Nn \__enumext_item_answer_diff:
1925 {
1926     \int_gset:Nn \g__enumext_item_answer_diff_int
1927     {
1928         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1929     }
1930 }

```

(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:`.

```

1931 \cs_new_protected:Nn \__enumext_check_ans_show:
1932 {
1933     \int_case:nn { \g__enumext_item_answer_diff_int }
1934     {
1935         { -1 }{ \__enumext_check_ans_msg_less: }
1936         { 0 }{ \__enumext_check_ans_msg_same_ok: }
1937         { 1 }{ \__enumext_check_ans_msg_greater: }
1938     }
1939 }

```

```

1940 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1941 {
1942   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1943   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1944 }
1945 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1946 {
1947   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1948   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1949 }
1950 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1951 {
1952   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1953   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1954 }

```

(End of definition for \\_\_enumext\_check\_ans\_show: and others.)

```

\__enumext_check_ans_log:
\__enumext_check_ans_log_msg_less:
\__enumext_check_ans_log_msg_same_ok:
\__enumext_check_ans_log_msg_greater:

```

The function \\_\_enumext\_check\_ans\_log: will be executed within the function \\_\_enumext\_execute\_after\_env: when the key `check-ans` is not active, that is, when `\g__enumext_check_ans_key_bool` is “false” and write in the log the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `\__enumext_item_answer_diff:`.

```

1955 \cs_new_protected:Nn \__enumext_check_ans_log:
1956 {
1957   \int_case:nn { \g__enumext_item_answer_diff_int }
1958   {
1959     { -1 } { \__enumext_check_ans_log_msg_less: }
1960     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1961     { 1 } { \__enumext_check_ans_log_msg_greater: }
1962   }
1963 }
1964 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1965 {
1966   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1967   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1968 }
1969 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1970 {
1971   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1972   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1973 }
1974 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1975 {
1976   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1977   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1978 }

```

(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.

```

1979 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1980 {
1981   \int_compare:nNnT
1982     { \g__enumext_check_starred_cmd_int } = { 0 }
1983     {
1984       \msg_warning:nnnV
1985         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1986     }
1987   \int_compare:nNnT
1988     { \g__enumext_check_starred_cmd_int } > { 1 }
1989     {
1990       \msg_warning:nnnV
1991         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1992     }
1993   \int_gzero:N \g__enumext_check_starred_cmd_int
1994   \tl_clear:N \l__enumext_check_start_line_env_tl
1995 }

```

(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*`.

```

1996 \cs_set_protected:Npn \__enumext_tmp:n #1
1997 {
1998   \keys_define:nn { enumext / #1 }
1999   {
2000     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2001     wrap-ans .initial:n =
2002       {
2003         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2004       },
2005     wrap-ans .value_required:n = true,
2006     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2007     wrap-opt .initial:n = [{##1}],
2008     wrap-opt .value_required:n = true,
2009     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2010     save-sep .initial:n = {, ~ },
2011     save-sep .value_required:n = true,
2012     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2013     mark-ans .initial:n = \textasteriskcentered,
2014     mark-ans .value_required:n = true,
2015     mark-pos .choice:,
2016     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2017     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2018     mark-pos / unknown .code:n =
2019       \msg_error:nnee { enumext } { unknown-choice }
2020       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2021     mark-pos .initial:n = right,
2022     mark-pos .value_required:n = true,
2023     show-ans .bool_set:N = \__enumext_show_answer_bool,
2024     show-ans .initial:n = false,
2025     show-ans .value_required:n = true,
2026     show-pos .bool_set:N = \__enumext_show_position_bool,
2027     show-pos .initial:n = false,
2028     show-pos .value_required:n = true,
2029     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2030     mark-ref .initial:n = \textasteriskcentered,
2031     mark-ref .value_required:n = true,
2032     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2033     save-ref .initial:n = false,
2034     save-ref .value_required:n = true,
2035   }
2036 }
2037 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for `wrap-ans` and others.)

For the `keyans` and `keyans*` environments we will only add the keys `mark-pos`, `show-ans` and `show-pos`.

```

2038 \cs_set_protected:Npn \__enumext_tmp:n #1
2039 {
2040   \keys_define:nn { enumext / #1 }
2041   {
2042     mark-pos .choice:,
2043     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2044     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2045     mark-pos .initial:n = right,
2046     mark-pos .value_required:n = true,
2047     show-ans .bool_set:N = \__enumext_show_answer_bool,
2048     show-ans .initial:n = false,
2049     show-ans .value_required:n = true,
2050     show-pos .bool_set:N = \__enumext_show_position_bool,
2051     show-pos .initial:n = false,
2052     show-pos .value_required:n = true,
2053   }
2054 }
2055 \clist_map_inline:nn { keyans, keyans* } { \__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 `__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `__enumext_store_save_key_X_tl` with the filtered *(keys)*.

```

2056 \cs_new_protected:Npn __enumext_store_active_keys:n #1
2057 {
2058   \bool_if:cF { __enumext_store_save_key_ __enumext_level: _bool }
2059   {
2060     \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2061     \tl_set:ce
2062       { __enumext_store_save_key_ __enumext_level: _tl }
2063       { __enumext_filter_save_key:n {#1} }
2064   }
2065 }
2066 \cs_new_protected:Npn __enumext_store_active_keys_vii:n #1
2067 {
2068   \bool_if:NF __enumext_store_save_key_vii_bool
2069   {
2070     \tl_clear:N __enumext_store_save_key_vii_tl
2071     \tl_set:Ne __enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2072   }
2073 }

```

(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.

```

2074 \cs_set_protected:Npn __enumext_tmp:n #1
2075 {
2076   \keys_define:nn { enumext / enumext* }
2077   {
2078     save-key .code:n = __enumext_parse_save_key_vii:n {##1},
2079     save-key .value_required:n = true,
2080   }
2081   \keys_define:nn { enumext / #1 }
2082   {
2083     save-key .code:n = __enumext_parse_save_key:n {##1},
2084     save-key .value_required:n = true,
2085   }
2086 }
2087 \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*`.

```

2088 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2089 {
2090   \bool_set_true:c { __enumext_store_save_key_ __enumext_level: _bool }
2091   \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2092   \tl_set:ce
2093     { __enumext_store_save_key_ __enumext_level: _tl }
2094     { __enumext_filter_save_key:n {#1} }
2095 }

```

```

2096 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2097 {
2098   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2099   \tl_clear:N \l__enumext_store_save_key_vii_tl
2100   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2101 }

```

(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

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

```

```

2102 \cs_new:Npn \__enumext_filter_save_key:n #1
2103 {
2104   \use:e
2105   {
2106     \keyval_parse:NNn
2107     \__enumext_filter_save_key_key:n
2108     \__enumext_filter_save_key_pair:nn {#1}
2109   }
2110 }

```

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.

```

2111 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2112 {
2113   \str_case:nnF {#1}
2114   {
2115     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2116   }
2117   { , { \exp_not:n {#1} } }
2118 }

```

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.

```

2119 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2120 {
2121   \str_case:nnF {#1}
2122   {
2123     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2124     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2125     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2126     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2127     { mini-right* } {}
2128   }
2129   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2130 }

```

(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 *\getkeysans* command.

The form in which the content is “stored” in the *⟨prop list⟩* is {*⟨position⟩*}{*⟨content⟩*}. This function is used by *\anskey* in *enumext* and *enumext\** environments, *\item\** in *keyans* and *keyans\** environments and *\anspic\** in *keyanspic* environment.

```

2131 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2132 {
2133   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2134   {
2135     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2136   }
2137   { #1 }
2138 }
2139 \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

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 an 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.

```

2140 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2141 {
2142   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2143 }
2144 \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.

```

2145 \cs_new_protected:Nn \__enumext_store_level_open:
2146 {
2147   \bool_if:NT \__enumext_check_answers_bool
2148   {
2149     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2150     {
2151       \__enumext_store_addto_seq:n
2152       {
2153         \item \begin{enumext}
2154       }
2155     }
2156     {
2157       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2158       {
2159         \item \begin{enumext} [
2160       }
2161       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2162       {
2163         ]
2164       }
2165       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2166     }
2167   }
2168 }
2169 \cs_new_protected:Nn \__enumext_store_level_close:
2170 {
2171   \bool_if:NT \__enumext_check_answers_bool
2172   {
2173     \__enumext_store_addto_seq:n { \end{enumext} }
2174   }
2175 }

```

(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.

```

2176 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2177 {
2178   \bool_if:NT \__enumext_check_answers_bool
2179   {
2180     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2181     {
2182       \__enumext_store_addto_seq:n
2183       {
2184         \item \begin{enumext*}
2185       }
2186     }
2187     {
2188       \tl_put_left:Nn l__enumext_store_save_key_vii_tl
2189       {
2190         \item \begin{enumext*}[
2191       }
2192       \tl_put_right:Nn l__enumext_store_save_key_vii_tl

```

```

2193         {
2194         }
2195     }
2196     \__enumext_store_addto_seq:V \__enumext_store_save_key_vii_tl
2197 }
2198 }
2199 }
2200 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2201 {
2202     \bool_if:NT \__enumext_check_answers_bool
2203     {
2204         \__enumext_store_addto_seq:n { \end{enumext*} }
2205     }
2206 }

```

(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

```

2207 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2208 {
2209     \mode_leave_vertical:
2210     \skip_horizontal:n { -\dim_use:N #2 }
2211     \makebox[0pt][ r ]
2212     {
2213         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2214         {
2215             \tl_use:N \l__enumext_mark_answer_sym_tl
2216         }
2217     }
2218     \skip_horizontal:n { \dim_use:N #2 }
2219 }
2220 \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.

```

2221 \cs_new_protected:Nn \__enumext_store_internal_ref:
2222 {
2223     \cs_set_protected:Npn \__enumext_tmp:n ##1
2224     {
2225         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2226         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2227         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2228         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2229     }
2230     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2231     \cs_set:Npn \__enumext_tmp:n ##1
2232     { . \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.

```

2233 \bool_lazy_all:nT
2234 {
2235     { \bool_if_p:N \g__enumext_starred_bool }
2236     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2237 }
2238 {
2239     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2240     { \tl_use:N \l__enumext_label_copy_vii_tl }
2241 }

```



```

2242 \bool_lazy_all:nT
2243 {
2244   { \bool_not_p:n { \g__enumext_standar_bool } }
2245   { \bool_if_p:N \l__enumext_standar_bool }
2246   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2247 }
2248 {
2249   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2250   {
2251     \tl_use:N \l__enumext_label_copy_vii_tl
2252     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2253   }
2254 }

```

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.

```

2255 \bool_lazy_all:nT
2256 {
2257   { \bool_if_p:N \g__enumext_standar_bool }
2258   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2259 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2260 }
2261 {
2262   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2263   {
2264     \tl_use:N \l__enumext_label_copy_i_tl
2265     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2266   }
2267 }
2268 \cs_set:Npn \__enumext_tmp:n ##1
2269 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2270 \bool_lazy_all:nT
2271 {
2272   { \bool_if_p:N \g__enumext_standar_bool }
2273   { \bool_if_p:N \l__enumext_starred_bool }
2274   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2275 }
2276 {
2277   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2278   {
2279     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2280     \tl_use:N \l__enumext_label_copy_vii_tl
2281   }
2282 }

```

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

```

2283 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2284 {
2285   \l__enumext_store_name_tl \c_colon_str
2286   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2287 }

```

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

```

2288 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2289 {
2290   \__enumext_newlabel:nn
2291   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2292   { \l__enumext_newlabel_arg_two_tl }
2293 }
2294 \l__enumext_write_aux_file_tl
2295 }

```

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

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

`\__enumext_store_anskey_code:n`

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

```

2296 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2297 {

```

```

2298 \int_gincr:N \g__enumext_item_anskey_int
2299 \__enumext_store_addto_prop:n {#1}
2300 \bool_if:NT \l__enumext_store_ref_key_bool
2301 {
2302   \__enumext_store_internal_ref:
2303 }
2304 \__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the [ $\langle key = val \rangle$ ] passed to the command to build our  $\backslash item$  in the variable  $\backslash l\_enumext\_store\_anskey\_arg\_tl$  which we will “store” in the  $\langle sequence \rangle$ . First we clear the variable  $\backslash l\_enumext\_store\_anskey\_arg\_tl$  and process the  $\langle keys \rangle$ , if the `break-col` key is present and the command is running under `enumext` (not in `enumext*`) we will add  $\backslash columnbreak$  and then  $\backslash item$ .

```

2305 \tl_clear:N \l__enumext_store_anskey_arg_tl
2306 \bool_lazy_and:nnT
2307 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2308 { \bool_not_p:n { \l__enumext_starred_bool } }
2309 {
2310   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2311 }
2312 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

If the `item-join` key is present and the command is running under `enumext*` we will add ( $\langle number \rangle$ ) to  $\backslash l\_enumext\_store\_anskey\_arg\_tl$ .

```

2313 \bool_lazy_and:nnT
2314 { \bool_not_p:n { \l__enumext_starred_bool } }
2315 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2316 {
2317   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2318   {
2319     ( \exp_not:V \l__enumext_store_item_join_int )
2320   }
2321 }

```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to  $\backslash l\_enumext\_store\_anskey\_arg\_tl$  along with the  $\langle argument \rangle$  for  $\backslash anskey$  or  $\langle body \rangle$  for `anskey*`.

```

2322 \bool_if:NTF \l__enumext_store_item_star_bool
2323 {
2324   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2325   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2326   {
2327     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2328     {
2329       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2330     }
2331   }
2332   \dim_compare:nT
2333   {
2334     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2335   }
2336   {
2337     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2338     {
2339       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2340     }
2341   }
2342   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2343 }
2344 {
2345   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2346 }

```

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  $\backslash hyperlink$  with `symbol` set by `mark-ref` key and then store in  $\langle sequence \rangle$ .

```

2347 \bool_lazy_and:nnT
2348 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2349 { \bool_if_p:N \l__enumext_hyperref_bool }
2350 {
2351   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2352   {
2353     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2354     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2355   }

```

```

2356     }
2357     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2358 }

```

(End of definition for \\_\_enumext\_store\_anskey\_code:n.)

\\_\_enumext\_anskey\_show\_wrap\_arg:n

The function \\_\_enumext\_anskey\_show\_wrap\_arg:n “wraps” the  $\langle argument \rangle$  passed to \anskey and the  $\langle body \rangle$  for anskey\* when using the wrap-ans key.

```

2359 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2360 {
2361     \par
2362     \bool_if:NTF \l__enumext_starred_bool
2363     {
2364         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2365     }
2366     {
2367         \__enumext_print_keyans_box:cc
2368         { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2369         { \l__enumext_labelsep_ \l__enumext_level: _dim }
2370     }
2371     \__enumext_anskey_wrapper:n { #1 }
2372 }

```

(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  $\langle prop list \rangle$  when using the show-pos key on the left margin next to the “wraps”  $\langle argument \rangle$  passed to \anskey and the  $\langle body \rangle$  in anskey\* on the right side when using the show-ans key.

```

2373 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2374 {
2375     \bool_if:NT \l__enumext_show_answer_bool
2376     {
2377         \__enumext_anskey_show_wrap_arg:n { #1 }
2378     }
2379     \bool_if:NT \l__enumext_show_position_bool
2380     {
2381         \tl_set:Nx \l__enumext_mark_answer_sym_tl
2382         {
2383             \group_begin:
2384             \exp_not:N \normalfont
2385             \exp_not:N \footnotesize [ \int_eval:n
2386             {
2387                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2388             }
2389             ]
2390             \group_end:
2391         }
2392         \__enumext_anskey_show_wrap_arg:n { #1 }
2393     }
2394 }

```

(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  $\langle sequences \rangle$  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[ $\langle key = val \rangle$ ]{ $\langle content \rangle$ }.

\\_\_enumext\_anskey\_unknown:n

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

\\_\_enumext\_anskey\_unknown:nn

```

2395 \keys_define:nn { enumext / anskey }
2396 {
2397     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2398     break-col .default:n = true,
2399     break-col .value_forbidden:n = true,
2400     item-join .int_set:N = \l__enumext_store_item_join_int,
2401     item-join .value_required:n = true,
2402     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2403     item-star .default:n = true,

```

```

2404     item-star .value_forbidden:n = true,
2405     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2406     item-sym* .value_required:n = true,
2407     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2408     item-pos* .value_required:n = true,
2409     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2410 }

```

The `<keys>` are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `\l__enumext_anskey_unknown:n`.

```

2411 \cs_new_protected:Npn \l__enumext_anskey_unknown:n #1
2412 {
2413     \exp_args:NV \l__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2414 }
2415 \cs_new_protected:Npn \l__enumext_anskey_unknown:nn #1 #2
2416 {
2417     \tl_if_blank:nTF {#2}
2418     {
2419         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2420     }
2421     {
2422         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2423     }
2424 }

```

(End of definition for `\l__enumext_anskey_unknown:n` and `\l__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 `\l__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 `\l__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 `\l__enumext_store_anskey_code:n`.

```

2425 \NewDocumentCommand \anskey { o +m }
2426 {
2427     \l__enumext_anskey_safe_outer:
2428     \group_begin:
2429         \bool_if:NT \l__enumext_check_answers_bool
2430         {
2431             \tl_if_novalue:nF {#1}
2432             {
2433                 \keys_set:nn { enumext / anskey } {#1}
2434             }
2435             \tl_if_blank:nTF {#2}
2436             {
2437                 \msg_error:nn { enumext } { anskey-empty-arg }
2438             }
2439             {
2440                 \l__enumext_anskey_safe_inner:
2441                 \l__enumext_store_anskey_code:n {#2}
2442             }
2443         }
2444     \group_end:
2445 }

```

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

### 12.29.1 Internal functions for the command

`\l__enumext_anskey_safe_outer:` The `\l__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.

`\l__enumext_anskey_safe_inner:`

```

2446 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2447 {
2448     \bool_if:NF \l__enumext_store_active_bool
2449     {
2450         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2451     }
2452     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2453     {
2454         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }

```

```

2455     }
2456     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2457     {
2458         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2459     }
2460     \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2461     {
2462         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2463     }
2464 }

```

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.

```

2465 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2466 {
2467     \int_incr:N \__enumext_anskey_level_int
2468     \int_compare:nNt { \__enumext_anskey_level_int } > { 1 }
2469     {
2470         \msg_error:nn { enumext } { anskey-nested }
2471     }
2472     \bool_if:NF \__enumext_item_number_bool
2473     {
2474         \msg_error:nn { enumext } { anskey-unnumber-item }
2475     }
2476     \mode_if_math:T
2477     {
2478         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2479     }
2480 }

```

(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.

```

2481 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2482 {
2483     \cs_undefine:c { anskey* }
2484     \cs_undefine:c { endanskey* }
2485     \cs_undefine:c { __scontents_anskey*_env_begin: }
2486     \cs_undefine:c { __scontents_anskey*_env_end: }
2487 }

```

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

```

2488 \__enumext_before_env:nn { enumext }
2489 {
2490     \bool_lazy_and:nnT
2491     { \int_compare_p:nNt { \__enumext_level_int } = { 0 } }
2492     { \int_compare_p:nNt { \__enumext_level_h_int } = { 0 } }
2493     {
2494         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2495         {
2496             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2497         }
2498     }

```

```

2499   }
2500   \__enumext_before_env:nn { enumext* }
2501   {
2502     \bool_lazy_and:nnT
2503     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2504     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2505     {
2506       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2507       {
2508         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2509       }
2510     }
2511   }

```

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.

```

2512   \__enumext_before_env:nn { anskey* }
2513   {
2514     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2515     {
2516       \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2517     }
2518     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2519     {
2520       \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2521     }
2522     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2523     {
2524       \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2525     }
2526     \bool_if:NF \__enumext_item_number_bool
2527     {
2528       \msg_error:nn { enumext } { anskey-unnumber-item }
2529     }
2530     \mode_if_math:T
2531     {
2532       \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2533     }
2534   }

```

(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`.

```

2535   \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2536   {
2537     \bool_if:NT \__enumext_anskey_env_bool
2538     {
2539       \newenvsc{anskey*}[store-env=#1,print-env=false]
2540       \__enumext_anskey_env_exec:
2541     }
2542   }
2543   \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`.

```

2544   \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2545   {
2546     \keys_define:nn { scontents / scontents }
2547     {
2548       break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2549       break-col .default:n = true,
2550       break-col .value_forbidden:n = true,
2551       item-join .int_gset:N = \g__enumext_store_item_join_int,
2552       item-join .value_required:n = true,

```

```

2553     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2554     item-star .default:n = true,
2555     item-star .value_forbidden:n = true,
2556     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2557     item-sym* .value_required:n = true,
2558     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2559     item-pos* .value_required:n = true,
2560     print-env .undefine:,
2561     store-env .undefine:,
2562     write-out .undefine:,
2563     unknown .code:n = { \__enumext_anskey_env_unknown:n {##1} },
2564   }
2565 }

```

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`.

```

2566 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2567 {
2568   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2569 }
2570 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2571 {
2572   \tl_if_blank:nTF {#2}
2573   {
2574     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2575   }
2576   {
2577     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2578   }
2579 }

```

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`.

```

2580 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2581 {
2582   \keys_define:nn { scontents / scontents }
2583   {
2584     break-col .undefine:,
2585     item-join .undefine:,
2586     item-star .undefine:,
2587     item-sym* .undefine:,
2588     item-pos* .undefine:,
2589     write-out .code:n = {
2590       \bool_set_false:N \l__scontents_storing_bool
2591       \bool_set_true:N \l__scontents_writing_bool
2592       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2593     },
2594     write-out .value_required:n = true,
2595     print-env .meta:nn = { scontents } { print-env = ##1 },
2596     print-env .default:n = true,
2597     store-env .meta:nn = { scontents } { store-env = ##1 },
2598     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2599   }
2600 }

```

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*.

```

2601 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2602 {
2603   \group_begin:
2604     \int_set:Nn \tex_newlinechar:D { `^^J }
2605     \__scontents_rescan_tokens:x
2606     {
2607       \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2608       #1
2609     }
2610 }

```

(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)*.

```

2611 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2612 {
2613   \__enumext_before_env:nn { anskey* }
2614   {
2615     \__enumext_anskey_env_define_keys:
2616   }

```

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.

```

2617   \hook_if_empty:nF {env/anskey*/after}
2618   {
2619     \hook_gremove_code:nn {env/anskey*/after} { * }
2620   }
2621   \__enumext_after_env:nn { anskey* }
2622   {
2623     \__enumext_anskey_env_save_keys:
2624     \tl_clear:N \l__enumext_store_anskey_env_tl
2625     \tl_clear:N \l__enumext_store_anskey_opt_tl
2626     \bool_if:NT \l__enumext_check_answers_bool
2627     {
2628       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2629       {
2630         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2631       }
2632       \regex_match:nVTF
2633       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2634       \l__enumext_store_anskey_env_tl
2635       {
2636         \msg_error:nn { enumext } { anskey-empty-arg }
2637       }
2638       {
2639         \__enumext_anskey_env_store:
2640       }
2641     }
2642     \__enumext_anskey_env_clean_vars:
2643     \__enumext_anskey_env_reset_keys:
2644   }
2645 }

```

• 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:` 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*.

`\__enumext_anskey_env_store:`

`\__enumext_anskey_env_clean_vars:`

```

2646 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2647 {
2648   \bool_lazy_and:nnT
2649   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2650   { \bool_not_p:n { \l__enumext_starred_bool } }
2651   {
2652     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2653   }

```

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

```

2654   \bool_lazy_and:nnT
2655   { \bool_not_p:n { \l__enumext_starred_bool } }
2656   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2657   {
2658     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2659     {
2660       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2661     }
2662   }

```

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

```

2663   \bool_if:NT \g__enumext_store_item_star_bool
2664   {
2665     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2666     {
2667       ,item-star,
2668     }
2669     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2670     {
2671       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2672       {
2673         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2674       }
2675     }
2676     \dim_compare:nT
2677     {
2678       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2679     }
2680     {
2681       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2682       {
2683         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2684       }
2685     }
2686   }
2687 }

```

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`.

```

2688 \cs_new_protected:Nn \__enumext_anskey_env_store:
2689 {
2690   \group_begin:
2691   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2692   {
2693     \exp_args:Ne
2694     \__enumext_store_anskey_code:n
2695     {
2696       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2697     }
2698   }
2699   {
2700     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2701     \exp_args:Ne
2702     \__enumext_store_anskey_code:n
2703     {
2704       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2705     }
2706   }
2707   \group_end:
2708 }

```

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

```

2709 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2710 {
2711   \bool_gset_false:N \g__enumext_store_columns_break_bool
2712   \int_gzero:N       \g__enumext_store_item_join_int
2713   \bool_gset_false:N \g__enumext_store_item_star_bool
2714   \tl_gclear:N       \g__enumext_store_item_symbol_tl
2715   \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2716 }

```

(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.

```

2717 \cs_new_protected:Nn \__enumext_execute_after_env:
2718 {
2719   \int_compare:nNt { \__enumext_level_int } = { 0 }
2720   {
2721     \tl_if_empty:NF \g__enumext_store_name_tl
2722     {
2723       \__enumext_stop_save_ans_msg:
2724       \__enumext_item_answer_diff:
2725       \__enumext_log_global_vars:
2726       \__enumext_log_answer_vars:
2727       \bool_if:NTF \g__enumext_check_ans_key_bool
2728       {
2729         \__enumext_check_ans_show:
2730       }
2731       { \__enumext_check_ans_log: }
2732       \__enumext_undefine_anskey_env:
2733     }
2734     \__enumext_reset_global_vars:
2735   }
2736 }

```

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

- This function is passed to the function `\__enumext_after_env:n` 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 `<label>` `\l__enumext_label_v_tl` for the `keyans` environment and the current `<label>` `\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 *<prop list>* defined by the `save-ans` key using the `\__enumext_store_addto_prop:V`.

```

2737 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2738 {
2739   \tl_clear:N \l__enumext_store_current_label_tl
2740   \int_compare:nNtF { \__enumext_keyans_pic_level_int } = { 1 }
2741   {
2742     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2743   }
2744   {
2745     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2746   }
2747   \tl_if_novalue:NF { #1 }
2748   {
2749     % Set save-sep
2750     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2751     {
2752       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2753     }
2754     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2755   }
2756   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2757 }

```

(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 `<label>`. 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 `<labels>` and remove the dots “.” from them, we do not want to get double dots in our references.

```

2758 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2759 {
2760   \bool_if:NT \l__enumext_store_ref_key_bool
2761   {
2762     \cs_set_protected:Npn \__enumext_tmp:n ##1
2763     {
2764       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2765       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2766       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2767       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2768     }
2769     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2770     \__enumext_keyans_store_ref_aux_i:
2771   }
2772 }

```

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`.

```

2773 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2774 {
2775   \bool_if:NT \g__enumext_starred_bool
2776   {
2777     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2778   }
2779   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2780   {
2781     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2782     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2783   }
2784   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2785   {
2786     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2787     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2788   }
2789   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2790   {
2791     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2792     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2793   }
2794   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2795   {
2796     \l__enumext_store_name_tl \c_colon_str
2797     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2798   }
2799   \__enumext_keyans_store_ref_aux_ii:
2800 }

```

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.

```

2801 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2802 {
2803   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2804   {
2805     \__enumext_newlabel:nn
2806     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2807     { \l__enumext_newlabel_arg_two_tl }
2808   }
2809   \l__enumext_write_aux_file_tl
2810 }

```

(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.

```

2811 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1

```

```

2812 {
2813   \tl_clear:N \l__enumext_store_current_label_tl
2814   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2815   {
2816     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2817   }
2818   {
2819     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2820   }
2821   \tl_if_novalue:nF { #1 }
2822   {
2823     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2824     {
2825       \tl_put_right:Ne \l__enumext_store_current_label_tl
2826       {
2827         \l__enumext_store_keyans_item_opt_sep_tl
2828       }
2829     }
2830     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2831   }
2832   \__enumext_keyans_addto_seq_link:
2833 }

```

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.

```

2834 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2835 {
2836   \bool_lazy_and:nnT
2837   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2838   { \bool_if_p:N \l__enumext_hyperref_bool }
2839   {
2840     \tl_put_right:Ne \l__enumext_store_current_label_tl
2841     {
2842       \hfill \exp_not:N \hyperlink
2843       {
2844         \exp_not:V \l__enumext_newlabel_arg_one_tl
2845       }
2846       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2847     }
2848   }
2849   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2850   \bool_if:NT \l__enumext_check_answers_bool
2851   {
2852     \int_gincr:N \g__enumext_item_anskey_int
2853   }
2854 }

```

(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.

```

2855 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2856 {
2857   \tl_if_novalue:nF { #1 }
2858   {
2859     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2860   }
2861   \bool_if:NT \l__enumext_show_answer_bool
2862   {
2863     \__enumext_keyans_show_ans:
2864   }
2865   \bool_if:NT \l__enumext_show_position_bool
2866   {

```

```

2867     \__enumext_keyans_show_pos:
2868   }
2869 }
2870 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2871 {
2872   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2873   {
2874     \bool_lazy_or:nnT
2875       { \bool_if_p:N \l__enumext_show_answer_bool }
2876       { \bool_if_p:N \l__enumext_show_position_bool }
2877       {
2878         \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2879       }
2880   }
2881 }
2882 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2883 {
2884   \bool_if:NT \l__enumext_starred_bool
2885   {
2886     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2887     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2888   }
2889   \tl_put_left:Nn \l__enumext_label_v_tl
2890   {
2891     \__enumext_print_keyans_box:NN
2892     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2893   }
2894 }
2895 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2896 {
2897   \bool_if:NT \l__enumext_starred_bool
2898   {
2899     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2900     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2901   }
2902   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2903   {
2904     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2905     {
2906       \group_begin:
2907       \exp_not:N \normalfont
2908       \exp_not:N \footnotesize [ \int_eval:n
2909         {
2910           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2911         }
2912       ]
2913       \group_end:
2914     }
2915   }
2916   {
2917     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2918     {
2919       \group_begin:
2920       \exp_not:N \normalfont
2921       \exp_not:N \footnotesize [ \int_eval:n
2922         {
2923           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2924         }
2925       ]
2926       \group_end:
2927     }
2928   }
2929   \tl_put_left:Nn \l__enumext_label_v_tl
2930   {
2931     \__enumext_print_keyans_box:NN
2932     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2933   }
2934 }

```

(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`.

```

2935 \cs_new_protected:Npn \__enumext_default_item:n #1
2936 {
2937   \tl_if_novalue:nTF {#1}
2938   {
2939     \bool_if:NT \l__enumext_check_answers_bool
2940     {
2941       \int_gincr:N \g__enumext_item_number_int
2942       \bool_set_true:N \l__enumext_item_number_bool
2943     }
2944     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2945     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2946   }
2947   {
2948     \bool_set_eq:cc
2949     { \l__enumext_wrap_label_ \__enumext_level: _bool }
2950     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2951     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2952   }
2953 }

```

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

`__enumext_starred_item:nn`

`__enumext_item_star_exec:`

The `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]` works like the *numbered* `\item`, but placing a `⟨symbol⟩` to the “left” of the `⟨label⟩` separated from it by the value the second optional argument `⟨offset⟩`.

`#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`.

```

2954 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2955 {
2956   \tl_if_novalue:nTF {#1}
2957   {
2958     \tl_gset_eq:Nc
2959     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2960   }
2961   {
2962     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
2963   }
2964   \tl_if_novalue:nTF {#2}
2965   {
2966     \dim_set_eq:cc
2967     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2968     { \l__enumext_labelsep_ \__enumext_level: _dim }
2969   }
2970   {
2971     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2972   }
2973   \bool_if:NT \l__enumext_check_answers_bool
2974   {
2975     \int_gincr:N \g__enumext_item_number_int
2976     \bool_set_true:N \l__enumext_item_number_bool

```



```

2977     }
2978     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2979     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2980 }

```

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

```

2981 \cs_new_protected:Nn \__enumext_item_star_exec:
2982 {
2983   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2984   {
2985     \mode_leave_vertical:
2986     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2987     \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_tl }
2988     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2989   }
2990 }

```

(End of definition for `\__enumext_starred_item:nn` and `\__enumext_item_star_exec:`.)

```

\__enumext_redefine_item:
\__enumext_make_label

```

The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`.

```

2991 \cs_new_protected:Nn \__enumext_redefine_item:
2992 {
2993   \RenewDocumentCommand \item { s o o }
2994   {
2995     \bool_if:nTF {##1}
2996     {
2997       \__enumext_starred_item:nn {##2} {##3}
2998     }
2999     { \__enumext_default_item:n {##2} }
3000   }
3001 }

```

The function `\__enumext_make_label:` redefine `\make_label` for the keys `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment.

```

3002 \cs_new_protected:Nn \__enumext_make_label:
3003 {
3004   \RenewDocumentCommand \make_label { m }
3005   {
3006     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3007     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3008     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3009     {
3010       \__enumext_item_star_exec:
3011       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3012     }
3013     { ##1 }
3014     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3015     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3016   }
3017 }

```

(End of definition for `\__enumext_redefine_item:` and `\__enumext_make_label:`.)

🔗 This 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*
3018 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3019 {
3020   \keys_define:nn { enumext / #1 }
3021   {
3022     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3023     item-sym* .value_required:n = true,
3024     item-sym* .initial:n = { $\star$ },
3025     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3026     item-pos* .value_required:n = true,
3027   }

```

```

3028     }
3029 \clist_map_inline:nn
3030 {
3031     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3032 }
3033 { \__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*`

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

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3034 \cs_set_protected:Npn \__enumext_tmp:n #1
3035 {
3036     \keys_define:nn { enumext / #1 }
3037     {
3038         unknown .code:n = { \__enumext_keyans_unknown_keys:n {#1} }
3039     }
3040 }
3041 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3042 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3043 {
3044     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3045 }
3046 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3047 {
3048     \tl_if_blank:nTF {#2}
3049     {
3050         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3051     }
3052     {
3053         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3054     }
3055 }

```

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

### 12.35.2 Handling unknown keys for `enumext*`

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

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3056 \keys_define:nn { enumext / enumext* }
3057 {
3058     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3059 }

```

Internal functions for handling `unknown` key.

```

3060 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3061 {
3062     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3063 }
3064 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3065 {
3066     \tl_if_blank:nTF {#2}
3067     {
3068         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3069     }
3070     {
3071         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3072     }
3073 }

```

(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.

```

3074 \cs_set_protected:Npn \__enumext_tmp:n #1
3075 {
3076   \keys_define:nn { enumext / #1 }
3077   {
3078     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3079   }
3080 }
3081 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3082 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3083 {
3084   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3085 }
3086 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3087 {
3088   \tl_if_blank:nTF {#2}
3089   {
3090     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3091   }
3092   {
3093     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3094   }
3095 }

```

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

### 12.36 Redefining `\item` and `\makeLabel` in keyans

The `\item` and `\item[⟨custom⟩]` commands work in the usual way in `keyans`, but the `\item*` and `\item*[⟨content⟩]` commands *store* the current `⟨label⟩` next to the `⟨content⟩` if it is present in the `⟨sequence⟩` and `⟨prop list⟩` defined by `save-ans` key.

`\__enumext_keyans_default_item:n` The function `\__enumext_keyans_default_item:n` executes the original behavior of the `\item`.

```

3096 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3097 {
3098   \tl_if_no_value:nTF { #1 }
3099   {
3100     \bool_set_true:N \__enumext_wrap_label_v_bool
3101     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3102   }
3103   {
3104     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3105     \__enumext_item_std:w [ #1 ] \tl_use:N \__enumext_fake_item_indent_v_tl
3106   }
3107 }

```

(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⟩`.

```

3108 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3109 {
3110   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3111   \__enumext_keyans_show_left:n { #1 }
3112   \bool_set_true:N \__enumext_wrap_label_v_bool
3113   \__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⟩`.

```

3114   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3115   \__enumext_keyans_addto_prop:n { #1 }
3116   \__enumext_keyans_store_ref:
3117   \__enumext_keyans_addto_seq:n { #1 }
3118   \int_gincr:N \g__enumext_check_starred_cmd_int
3119 }

```

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

`\item*` 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.

```

3120 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3121 {
3122   \RenewDocumentCommand \item { s o }
3123   {
3124     \bool_if:nTF {##1}
3125     {
3126       \peek_remove_spaces:n
3127       {
3128         \__enumext_keyans_starred_item:n {##2}
3129       }
3130     }
3131     {
3132       \__enumext_keyans_default_item:n {##2}
3133     }
3134   }
3135 }

```

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

```

3136 \cs_new_protected:Nn \__enumext_keyans_make_label:
3137 {
3138   \RenewDocumentCommand \makeLabel { m }
3139   {
3140     \tl_use:N \l__enumext_label_fill_left_v_tl
3141     \tl_use:N \l__enumext_label_font_style_v_tl
3142     \bool_if:nTF \l__enumext_wrap_label_v_bool
3143     {
3144       \__enumext_wrapper_label_v:n { ##1 }
3145     }
3146     { ##1 }
3147     \tl_use:N \l__enumext_label_fill_right_v_tl
3148   }
3149 }

```

(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.

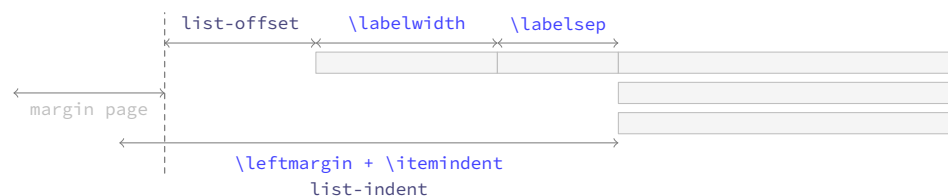


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

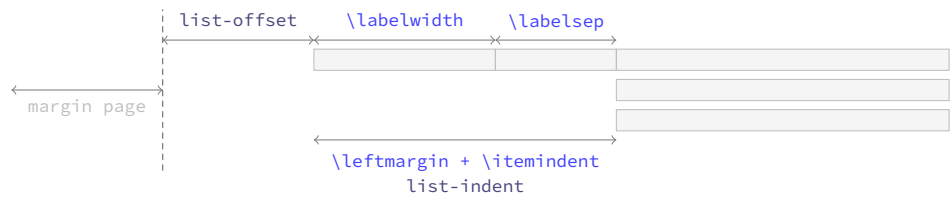
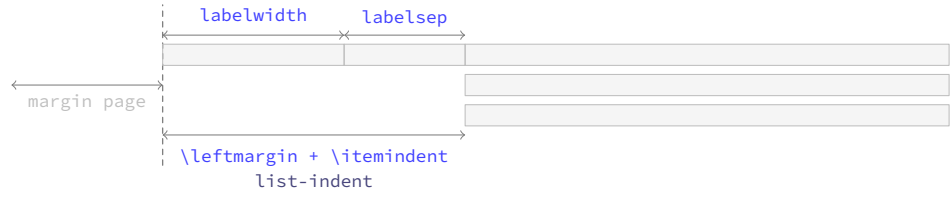
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

Figure 10: Representation of horizontal lengths concept in list in `enumext`.Figure 11: Default horizontal lengths in `enumext`.

```

#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).

```

3150 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3151 {
3152   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3153   {
3154     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3155     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3156   }
3157   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3158   {
3159     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3160     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3161   }

```

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

```

3162   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3163   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3164   {
3165     \dim_set:Nn #6 { #1 + #2 - #4 }
3166     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3167   }
3168   {
3169     \dim_compare:nNnT { #4 } = { #1 + #2 }
3170     { \dim_set:Nn #6 { \c_zero_dim } }
3171     \dim_compare:nNnT { #4 } < { #1 + #2 }
3172     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3173     \dim_compare:nNnT { #4 } > { #1 + #2 }
3174     {
3175       \dim_set:Nn #6 { -#1 - #2 + #4 }
3176       \dim_set:Nn #6 { #6*-1 }
3177     }
3178     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3179   }
3180 }
3181 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNN { ccccccc }

```

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

### 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.

```

3182 \cs_set_protected:Npn \__enumext_tmp:n #1
3183 {
3184   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3185   {
3186     \__enumext_calc_hspace:ccccc
3187     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3188     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3189     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3190     { \__enumext_leftmargin_tmp_#1_bool }
3191     \clist_map_inline:nn
3192       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3193       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3194     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3195       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3196     \usecounter { enumX#1 }
3197     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3198     \str_if_eq:nnTF {#1} { v }
3199     {
3200       \__enumext_keyans_redefine_item:
3201       \__enumext_keyans_make_label:
3202       \__enumext_keyans_ref:
3203       \__enumext_keyans_fake_item:
3204       \bool_if:cT { \__enumext_show_length_#1_bool }
3205       {
3206         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3207       }
3208     }
3209     {
3210       \__enumext_redefine_item:
3211       \__enumext_make_label:
3212       \__enumext_standar_ref:
3213       \__enumext_fake_item:
3214       \bool_if:cT { \__enumext_show_length_#1_bool }
3215       {
3216         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
3217       }
3218     }
3219   }
3220 }
3221 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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 `\parskip` locally.

```

3222 \cs_set_protected:Npn \__enumext_tmp:n #1
3223 {
3224   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3225   {
3226     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3227     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3228     \__enumext_calc_hspace:ccccc
3229     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3230     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3231     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3232     { \__enumext_leftmargin_tmp_#1_bool }
3233     \clist_map_inline:nn
3234       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3235       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3236     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3237       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3238     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3239     \skip_zero:N \partopsep
3240     \usecounter { enumX#1 }
3241     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```

```

3242     \__enumext_starred_ref:
3243     \str_if_eq:nnTF {#1} { vii }
3244     {
3245         \__enumext_fake_item_vii:
3246         \bool_if:cT { \__enumext_show_length_vii_bool }
3247         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3248     }
3249     {
3250         \__enumext_fake_item_viii:
3251         \bool_if:cT { \__enumext_show_length_#1_bool }
3252         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3253     }
3254 }
3255 }
3256 \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.

```

3257 \NewDocumentEnvironment{enumext}{ 0{} }
3258 {
3259     \__enumext_safe_exec:
3260     \__enumext_parse_keys:n {#1}
3261     \__enumext_before_list:
3262     \__enumext_start_store_level:
3263     \__enumext_start_list:nn
3264     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3265     {
3266         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3267         \__enumext_before_keys_exec:
3268     }
3269     \__enumext_set_item_width:
3270     \__enumext_after_args_exec:
3271 }
3272 {
3273     \__enumext_stop_list:
3274     \__enumext_stop_store_level:
3275     \__enumext_after_list:
3276 }

```

(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.

```

3277 \cs_new_protected:Nn \__enumext_set_item_width:
3278 {
3279     \dim_set:Nn \itemwidth
3280     {
3281         \linewidth
3282     }
3283     \dim_compare:nT
3284     {
3285         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3286     }
3287     {
3288         \dim_sub:Nn \itemwidth
3289         {
3290             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3291         }
3292     }
3293 }

```

(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 **\\_\_enumext\_level\_int** to restrict nesting of the environment, set **\\_\_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”.

```

3294 \cs_new_protected:Nn \__enumext_safe_exec:
3295 {
3296   \__enumext_internal_mini_page:
3297   \__enumext_is_not_nested:
3298   \int_incr:N \__enumext_level_int
3299   \int_compare:nNt { \__enumext_level_int } > { 4 }
3300   { \msg_fatal:nn { enumext } { list-too-deep } }
3301   \bool_set_true:N \__enumext_standar_bool
3302   \bool_set_false:N \__enumext_starred_bool
3303   \__enumext_is_on_first_level:
3304 }

```

(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.

```

3305 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3306 {
3307   \tl_if_novalue:nF {#1}
3308   {
3309     \str_clear:N \l__enumext_series_str
3310     \int_compare:nNtF { \__enumext_level_int } = { 1 }
3311     {
3312       \keys_set:nn { enumext / level-1 } {#1}
3313       \__enumext_parse_series:n {#1}
3314       \__enumext_nested_base_line_fix:
3315     }
3316     {
3317       \exp_args:Ne \keys_set:nn
3318       { enumext / level-\int_use:N \__enumext_level_int } {#1}
3319     }
3320     \__enumext_store_active_keys:n {#1}
3321   }
3322 }

```

(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*`.

```

3323 \cs_new_protected:Nn \__enumext_start_store_level:
3324 {
3325   \bool_lazy_all:nT
3326   {
3327     { \bool_if_p:N \__enumext_store_active_bool }
3328     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3329     { \bool_if_p:N \g__enumext_standar_bool }
3330   }
3331   {
3332     \int_compare:nNt { \__enumext_level_int } > { 1 }
3333     {
3334       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3335       \__enumext_store_level_open:
3336     }
3337   }

```

If `enumext` are nested in `enumext*` add `\__enumext_store_level_open:` to preserve the stored structure.

```

3338   \bool_lazy_all:nT
3339   {
3340     { \bool_if_p:N \__enumext_store_active_bool }
3341     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3342     { \int_compare_p:nNt { \__enumext_level_h_int } = { 1 } }
3343   }
3344   {
3345     \int_compare:nNt { \__enumext_level_int } > { 0 }
3346     {

```

```

3347         \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3348         \__enumext_store_level_open:
3349     }
3350 }
3351 }

```

Close the stored structure.

```

3352 \cs_new_protected:Nn \__enumext_stop_store_level:
3353 {
3354     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3355     {
3356         \__enumext_store_level_close:
3357     }
3358 }

```

(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.

```

3359 \cs_new_protected:Nn \__enumext_before_list:
3360 {
3361     \__enumext_vspace_above:
3362     \__enumext_before_args_exec:
3363     \__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.

```

3364     \dim_compare:nNtT
3365     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3366     {
3367         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3368         {
3369             \linewidth
3370             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3371             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3372         }

```

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.

```

3373         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3374         \int_gincr:N \g__enumext_minipage_stat_int
3375         \__enumext_minipage_add_space:
3376         \begin{\__enumext_mini_env*}
3377         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3378     }
3379     \__enumext_multicols_start:
3380 }

```

(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.

```

3381 \cs_new_protected:Nn \__enumext_multicols_start:
3382 {
3383     \int_compare:nNtT
3384     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3385     {
3386         \dim_compare:nNtT
3387         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3388         {
3389             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3390             {

```

```

3391         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3392         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3393         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3394         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3395     }
3396 }
3397 \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3398 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3399 {
3400     \dim_zero:N \columnseprule
3401 }

```

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.

```

3402     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3403     {
3404         \skip_zero:N \multicolsep
3405         \__enumext_multi_addvspace:
3406     }
3407     \raggedcolumns
3408     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3409 }
3410 }

```

(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.

```

3411 \cs_new_protected:Nn \__enumext_multicols_stop:
3412 {
3413     \int_compare:nNt
3414     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3415     {
3416         \end{multicols}
3417         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3418         {
3419             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3420         }
3421     }
3422 }

```

(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.

```

3423 \cs_new_protected:Nn \__enumext_after_list:
3424 {
3425     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3426     {
3427         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3428         {
3429             \msg_warning:nn { enumext } { missing-miniright }
3430             \miniright
3431         }
3432         \int_gzero:N \g__enumext_minipage_stat_int
3433         \unskip % remove topsep + [partopsep]
3434         \end{\__enumext_mini_env*}
3435     }
3436     {
3437         \__enumext_multicols_stop:
3438     }

```

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.

```

3439     \__enumext_after_stop_list:

```

```

3440     \__enumext_check_ans_key_hook:
3441     \__enumext_vspace_below:
3442     \bool_set_false:N \l__enumext_standar_bool
3443     \__enumext_resume_save_counter:
3444 }

```

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`.

```

3445 \__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.

```

3446 \NewDocumentEnvironment{keyans}{0}{}
3447 {
3448     \__enumext_keyans_safe_exec:
3449     \__enumext_keyans_parse_keys:n {#1}
3450     \__enumext_before_list_v:
3451     \__enumext_start_list:nn
3452     { \tl_use:N \l__enumext_label_v_tl }
3453     {
3454         \__enumext_list_arg_two_v:
3455         \__enumext_before_keys_exec_v:
3456     }
3457     \__enumext_keyans_set_item_width:
3458     \__enumext_after_args_exec_v:
3459 }
3460 {
3461     \__enumext_check_starred_cmd:n { item }
3462     \__enumext_stop_list:
3463     \__enumext_after_list_v:
3464 }

```

(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.

```

3465 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3466 {
3467     \dim_set:Nn \itemwidth
3468     {
3469         \linewidth
3470     }
3471     \dim_compare:nT
3472     {
3473         \l__enumext_listoffset_v_dim != \c_zero_dim
3474     }
3475     {
3476         \dim_sub:Nn \itemwidth
3477         {
3478             \l__enumext_listoffset_v_dim
3479         }
3480     }
3481 }

```

(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.

```

3482 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3483 {
3484     \bool_if:NF \l__enumext_store_active_bool
3485     {
3486         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3487     }

```

```

3488 \int_incr:N \l__enumext_keyans_level_int
3489 \bool_set_true:N \l__enumext_keyans_env_bool
3490 \__enumext_keyans_name_and_start:
3491 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3492 \bool_set_false:N \l__enumext_store_active_bool
3493 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3494 {
3495   \msg_error:nn { enumext } { keyans-nested }
3496 }
3497 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3498 {
3499   \msg_error:nn { enumext } { keyans-wrong-level }
3500 }
3501 }

```

(End of definition for `\__enumext_keyans_safe_exec:`)

```

\__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
3502 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3503 {
3504   \keys_set:nn { enumext / keyans } {#1}
3505 }

```

(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:
3506 \cs_new_protected:Nn \__enumext_before_list_v:
3507 {
3508   \__enumext_vspace_above_v:
3509   \__enumext_before_args_exec_v:
3510   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3511   {
3512     \dim_set:Nn \l__enumext_minipage_left_v_dim
3513     {
3514       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3515     }
3516     \bool_set_true:N \l__enumext_minipage_active_v_bool
3517     \int_gincr:N \g__enumext_minipage_stat_int
3518     \__enumext_keyans_minipage_add_space:
3519     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3520   }
3521   \__enumext_keyans_multicols_start:
3522 }
3523 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3524 {
3525   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3526   {
3527     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3528     {
3529       \dim_set:Nn \l__enumext_columns_sep_v_dim
3530       {
3531         (
3532           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3533         ) / \l__enumext_columns_v_int
3534         - \l__enumext_listoffset_v_dim
3535       }
3536     }
3537     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3538     \dim_zero:N \columnseprule % no rule here
3539     \bool_if:NF \l__enumext_minipage_active_v_bool
3540     {
3541       \skip_zero:N \multicolsep
3542       \__enumext_keyans_multi_addvspace:
3543     }
3544     \raggedcolumns
3545     \begin{multicols}{ \l__enumext_columns_v_int }
3546   }
3547 }
3548 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3549 {
3550   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }

```

```

3551     {
3552         \end{multicols}
3553         \bool_if:NF \l__enumext_minipage_active_v_bool
3554         {
3555             \par\addvspace{ \l__enumext_multicols_below_v_skip }
3556         }
3557     }
3558 }
3559 \cs_new_protected:Nn \__enumext_after_list_v:
3560 {
3561     \bool_if:NTF \l__enumext_minipage_active_v_bool
3562     {
3563         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3564         {
3565             \msg_warning:nn { enumext } { missing-miniright }
3566             \miniright
3567         }
3568         \int_gzero:N \g__enumext_minipage_stat_int
3569         \unskip % remove topsep + [partopsep]
3570         \end{__enumext_mini_env*}
3571         \par\addvspace{ \l__enumext_minipage_after_skip }
3572     }
3573     {
3574         \__enumext_keyans_multicols_stop:
3575     }
3576     \bool_set_false:N \l__enumext_keyans_env_bool
3577     \__enumext_after_stop_list_v:
3578     \__enumext_vspace_below_v:
3579 }

```

(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.

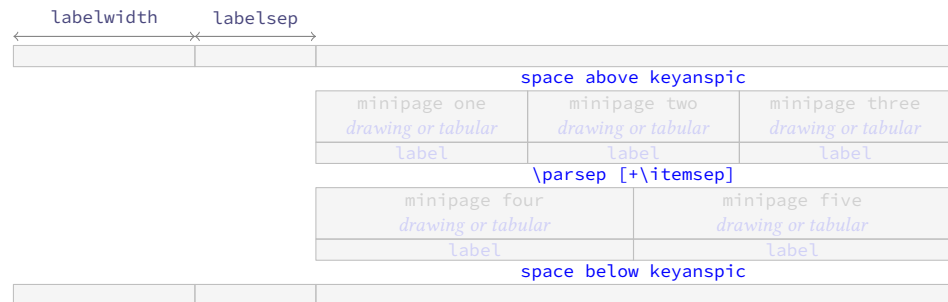


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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 takes 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.

```

3580 \NewDocumentCommand \anspic { s o +m }
3581 {

```

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

```

3582     \bool_if:NF \l__enumext_store_active_bool
3583     {
3584         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3585     }
3586     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3587     {

```

```

3588     \msg_error:nn { enumext } { keyanspic-wrong-level }
3589   }
3590   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3591   {
3592     \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3593   }

```

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.

```

3594   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3595   {
3596     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3597   }
3598 }

```

(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.

```

3599 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3600 {
3601   \stepcounter { enumXvi }
3602   #3 \l
3603   \bool_if:nT { #1 }
3604   {
3605     \__enumext_keyans_addto_prop:n { #2 }
3606     \__enumext_keyans_store_ref:
3607     \__enumext_keyans_addto_seq:n { #2 }
3608     \int_gincr:N \g__enumext_check_starred_cmd_int
3609     \bool_lazy_or:nnT
3610     { \bool_if_p:N \l__enumext_show_answer_bool }
3611     { \bool_if_p:N \l__enumext_show_position_bool }
3612     {
3613       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3614       \__enumext_keyans_show_left:n { #2 }
3615       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3616     }
3617   }
3618   \tl_use:N \l__enumext_label_font_style_v_tl
3619   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3620 }

```

(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.

```

3621 \NewDocumentEnvironment{keyanspic}{o}
3622 {
3623   \__enumext_keyans_pic_safe_exec:
3624   \__enumext_start_list:nn
3625   { }
3626   {
3627     \__enumext_keyans_pic_arg_two:
3628   }

```

We apply the “adjusted” vertical spacing above the environment

```

3629   \vspace { \l__enumext_keyans_pic_above_skip }
3630 }

```

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.

```

3631 {
3632   \tl_if_novalue:nTF { #1 }
3633   {
3634     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3635   }
3636   { \__enumext_keyans_pic_do:n { #1 } }
3637   \__enumext_stop_list:

```



```

3638     \__enumext_check_starred_cmd:n { ansPIC }
3639     \setcounter { enumXvi } { 0 }
3640     \vspace { \__enumext_topsep_v_skip }
3641     %\bool_set_false:N \__enumext_store_active_bool
3642 }

```

(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.

```

3643 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3644 {
3645     \int_incr:N \__enumext_keyans_pic_level_int
3646     \int_compare:nNtT { \__enumext_keyans_pic_level_int } > { 1 }
3647     {
3648         \msg_error:nn { enumext } { keyanspic-nested }
3649     }
3650     \__enumext_keyans_name_and_start:
3651 }

```

(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`.

```

3652 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3653 {
3654     \dim_compare:nNtT { #1 } < { 0pt }
3655     { \skip_set:Nn #1 { -#1 } }
3656 }

```

(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.

```

3657 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3658 {

```

The first thing to do is to set the boolean variable `\__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.

```

3659     \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
3660     \__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.

```

3661     \skip_add:Nn \parsep { \itemsep }
3662     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3663     \dim_zero:N \labelwidth
3664     \dim_zero:N \listparindent
3665     \dim_zero:N \labelsep
3666     \skip_zero:N \partopsep
3667     \skip_zero:N \itemsep

```

We set the value of `\__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.

```

3668     \__enumext_keyans_pic_skip_abs:N \parsep
3669     \skip_set:Nn \__enumext_keyans_pic_above_skip
3670     {
3671         \box_dp:N \strutbox
3672         + \__enumext_topsep_v_skip
3673         - \parsep
3674     }
3675     \__enumext_item_std:w \scan_stop:
3676     % paranoia
3677     \RenewDocumentCommand \item {}
3678     {
3679         \msg_error:nn { enumext } { keyanspic-item-cmd }
3680     }
3681 }

```

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

```
\__enumext_keyans_pic_do:n
\__enumext_keyans_pic_do:e
```

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`.

```
3682 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3683 {
3684   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3685 }
3686 \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 `\l__enumext_keyans_pic_body_seq` sequence inside them.

```
3687 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3688 {
3689   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3690   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3691   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3692   \int_step_inline:nnn
3693     { \l__enumext_keyans_pic_above_int + 1 }
3694     { \l__enumext_keyans_pic_below_int }
3695     {
3696       \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3697       \centering
3698       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3699       \__enumext_endminipage:
3700     }
3701   \par
3702 }
```

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

## 12.41 The horizontal environments

Generating horizontal list environments is NOT as simple as standard  $\text{\LaTeX}$  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](#).

```
3703 \cs_new_protected:Nn \__enumext_footnotetext:nn
3704 {
3705   \footnotetext[#1]{#2}
3706 }
3707 \cs_new_protected:Nn \__enumext_renew_footnote:
3708 {
3709   \seq_gclear:N \g__enumext_footnote_arg_seq
3710   \seq_gclear:N \g__enumext_footnote_int_seq
3711   \RenewDocumentCommand \footnote { o +m }
3712   {
3713     \tl_if_novalue:nTF {##1}
3714     {
3715       \stepcounter{footnote}
3716       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3717     }
3718     {
3719       \int_gset:Nn \g__enumext_footnote_int { ##1 }
3720     }
3721     \footnotemark [ \g__enumext_footnote_int ]
3722     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3723     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3724   }
3725 }
3726 \cs_new_protected:Nn \__enumext_print_footnote:
3727 {
3728   \seq_if_empty:NF \g__enumext_footnote_int_seq
3729   {
```

```

3730         \seq_map_pairwise_function:NNN
3731         \g__enumext_footnote_int_seq
3732         \g__enumext_footnote_arg_seq
3733         \__enumext_footnotetext:nn
3734     }
3735 }

```

(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`).

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:

```

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

```

3736 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3737 {
3738     \dim_compare:nNtT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3739     {
3740         \dim_set:Nn \__enumext_columns_sep_vii_dim
3741         {
3742             ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
3743             / \__enumext_columns_vii_int
3744         }
3745     }
3746     \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
3747     \dim_set:Nn \__enumext_item_width_vii_dim
3748     {
3749         ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
3750         / \__enumext_columns_vii_int
3751         - \__enumext_labelwidth_vii_dim
3752         - \__enumext_labelsep_vii_dim
3753     }

```

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

```

3754     \dim_compare:nNtT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3755     {
3756         \dim_sub:Nn \__enumext_item_width_vii_dim
3757         {
3758             ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
3759             / \__enumext_columns_vii_int
3760         }
3761         \dim_add:Nn \__enumext_columns_sep_vii_dim
3762         {
3763             \__enumext_rightmargin_vii_dim
3764         }
3765     }
3766 }

```

Same implementation for the `keyans*` environment.

```

3767 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3768 {
3769     \dim_compare:nNtT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3770     {
3771         \dim_set:Nn \__enumext_columns_sep_viii_dim
3772         {
3773             ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
3774             / \__enumext_columns_viii_int
3775         }
3776     }
3777     \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
3778     \dim_set:Nn \__enumext_item_width_viii_dim
3779     {
3780         ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
3781         / \__enumext_columns_viii_int
3782         - \__enumext_labelwidth_viii_dim
3783         - \__enumext_labelsep_viii_dim
3784     }
3785     \dim_compare:nNtT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3786     {

```

```

3787         \dim_sub:Nn \l__enumext_item_width_viii_dim
3788         {
3789             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3790             / \l__enumext_columns_viii_int
3791         }
3792     \dim_add:Nn \l__enumext_columns_sep_viii_dim
3793     {
3794         \l__enumext_rightmargin_viii_dim
3795     }
3796 }
3797 }

```

(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.

```

3798 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3799 {
3800     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3801     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3802     {
3803         \msg_warning:nnee { enumext } { item-joined }
3804         { \int_use:N \l__enumext_joined_item_vii_int }
3805         { \int_use:N \l__enumext_columns_vii_int }
3806         \int_set:Nn \l__enumext_joined_item_vii_int
3807         {
3808             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3809         }
3810     }
3811     \int_compare:nNnT
3812     { \l__enumext_joined_item_vii_int }
3813     >
3814     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3815     {
3816         \msg_warning:nnee { enumext } { item-joined-columns }
3817         { \int_use:N \l__enumext_joined_item_vii_int }
3818         {
3819             \int_eval:n
3820             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3821         }
3822         \int_set:Nn \l__enumext_joined_item_vii_int
3823         {
3824             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3825         }
3826     }
3827     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3828     {
3829         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3830         \int_decr:N \l__enumext_joined_item_aux_vii_int
3831         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3832         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3833         \dim_set:Nn \l__enumext_joined_width_vii_dim
3834         {
3835             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3836             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3837               + \l__enumext_columns_sep_vii_dim
3838               ) * \l__enumext_joined_item_aux_vii_int
3839         }
3840         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3841     }
3842     {
3843         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3844         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3845     }
3846 }

```

Same implementation for the `keyans*` environment.

```

3847 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3848 {

```

```

3849 \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3850 \int_compare:nNtT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3851 {
3852   \msg_warning:nnee { enumext } { item-joined }
3853   { \int_use:N \l__enumext_joined_item_viii_int }
3854   { \int_use:N \l__enumext_columns_viii_int }
3855   \int_set:Nn \l__enumext_joined_item_viii_int
3856   {
3857     \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3858   }
3859 }
3860 \int_compare:nNtT
3861 { \l__enumext_joined_item_viii_int }
3862 >
3863 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3864 {
3865   \msg_warning:nnee { enumext } { item-joined-columns }
3866   { \int_use:N \l__enumext_joined_item_viii_int }
3867   {
3868     \int_eval:n
3869     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3870   }
3871   \int_set:Nn \l__enumext_joined_item_viii_int
3872   {
3873     \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3874   }
3875 }
3876 \int_compare:nNtTF { \l__enumext_joined_item_viii_int } > { 1 }
3877 {
3878   \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3879   \int_decr:N \l__enumext_joined_item_aux_viii_int
3880   \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3881   \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3882   \dim_set:Nn \l__enumext_joined_width_viii_dim
3883   {
3884     \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3885     + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3886         + \l__enumext_columns_sep_viii_dim
3887     ) * \l__enumext_joined_item_aux_viii_int
3888   }
3889   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3890 }
3891 {
3892   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3893   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3894 }
3895 }

```

(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

```

\__enumext_start_mini_vii:
\__enumext_stop_mini_vii:

```

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `\__enumext_mini_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`.

```

3896 \cs_new_protected:Nn \__enumext_start_mini_vii:
3897 {
3898   \dim_compare:nNtT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3899   {
3900     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3901     {
3902       \linewidth
3903       - \l__enumext_minipage_right_vii_dim
3904       - \l__enumext_minipage_hsep_vii_dim
3905     }
3906     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3907     \dim_gset_eq:NN
3908     \g__enumext_minipage_right_vii_dim
3909     \l__enumext_minipage_right_vii_dim
3910     \__enumext_mini_addvspace_vii:
3911     \nointerlineskip\nindent

```

```

3912     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3913   }
3914 }

```

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:n` to execute the `__enumext_mini_env*` on the “right side”.

```

3915 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3916 {
3917   \bool_if:NT \l__enumext_minipage_active_vii_bool
3918   {
3919     \end{__enumext_mini_env*}
3920     \hfill
3921     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3922   }
3923 }

```

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.

```

3924 \__enumext_after_env:n {enumext*}
3925 {
3926   \bool_if:NT \g__enumext_minipage_active_vii_bool
3927   {
3928     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3929     \par\addvspace { \g__enumext_minipage_right_skip }
3930     \bool_if:NF \g__enumext_minipage_center_vii_bool
3931     {
3932       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3933       {
3934         \centering
3935       }
3936     }
3937     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3938     {
3939       \tl_use:N \g__enumext_miniright_code_vii_tl
3940     }
3941     \box_use_drop:N \l__enumext_miniright_code_vii_box
3942     \end{__enumext_mini_env*}
3943     \par\addvspace{ \g__enumext_minipage_after_skip }
3944   }
3945   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3946   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3947   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3948   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3949   \bool_gset_false:N \g__enumext_starred_bool
3950 }

```

(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.

```

3951 \cs_new_protected:Nn \__enumext_start_mini_viii:
3952 {
3953   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3954   {
3955     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3956     {
3957       \linewidth
3958       - \l__enumext_minipage_right_viii_dim
3959       - \l__enumext_minipage_hsep_viii_dim
3960     }
3961     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3962     \dim_gset_eq:NN
3963     \g__enumext_minipage_right_viii_dim
3964     \l__enumext_minipage_right_viii_dim
3965     \__enumext_mini_addvspace_viii:
3966     \nointerlineskip\noindent
3967     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }

```

```

3968     }
3969   }
3970   \cs_new_protected:Nn \__enumext_stop_mini_viii:
3971   {
3972     \bool_if:NT \l__enumext_minipage_active_viii_bool
3973     {
3974       \end{__enumext_mini_env*}
3975       \hfill
3976       \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3977     }
3978   }
3979   \__enumext_after_env:nn {keyans*}
3980   {
3981     \bool_if:NT \g__enumext_minipage_active_viii_bool
3982     {
3983       \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3984       \par\addvspace { \g__enumext_minipage_right_skip }
3985       \bool_if:NF \g__enumext_minipage_center_viii_bool
3986       {
3987         \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3988         {
3989           \centering
3990         }
3991       }
3992       \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3993       {
3994         \tl_use:N \g__enumext_miniright_code_viii_tl
3995       }
3996       \box_use_drop:N \l__enumext_miniright_code_viii_box
3997       \end{__enumext_mini_env*}
3998       \par\addvspace{ \g__enumext_minipage_after_skip }
3999     }
4000     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4001     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4002     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4003     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4004   }

```

(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.

```

4005 \NewDocumentEnvironment{enumext*}{ o }
4006 {
4007   \__enumext_safe_exec_vii:
4008   \__enumext_parse_keys_vii:n {#1}
4009   \__enumext_before_list_vii:
4010   \__enumext_start_store_level_vii:
4011   \__enumext_start_list:nn { }
4012   {
4013     \__enumext_list_arg_two_vii:
4014     \__enumext_before_keys_exec_vii:
4015   }
4016   \__enumext_starred_columns_set_vii:
4017   \item[] \scan_stop:
4018   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4019   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4020 }
4021 {
4022   \__enumext_stop_item_tmp_vii:
4023   \__enumext_remove_extra_parsep_vii:
4024   \__enumext_stop_list:
4025   \__enumext_stop_store_level_vii:
4026   \__enumext_after_list_vii:
4027 }

```

(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.

```

4028 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4029 {
4030   \__enumext_internal_mini_page:
4031   \__enumext_is_not_nested:
4032   \int_incr:N \l__enumext_level_h_int
4033   \int_compare:nNtT { \l__enumext_level_h_int } > { 1 }
4034   {
4035     \msg_error:nn { enumext } { nested }
4036   }
4037   \int_compare:nNtT { \l__enumext_keyans_level_h_int } = { 1 }
4038   {
4039     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4040   }
4041   \bool_set_true:N \l__enumext_starred_bool
4042   \bool_set_false:N \l__enumext_standar_bool
4043   \__enumext_is_on_first_level:
4044 }

```

(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`.

```

4045 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4046 {
4047   \tl_if_novalue:nF {#1}
4048   {
4049     \str_clear:N \l__enumext_series_str
4050     \keys_set:nn { enumext / enumext* } {#1}
4051     \__enumext_parse_series:n {#1}
4052     \__enumext_store_active_keys_vii:n {#1}
4053     \__enumext_nested_base_line_fix:
4054   }
4055 }

```

(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*`.

```

4056 \cs_new_protected:Nn \__enumext_before_list_vii:
4057 {
4058   \__enumext_vspace_above_vii:
4059   \__enumext_check_ans_active:
4060   \__enumext_before_args_exec_vii:
4061   \__enumext_start_mini_vii:
4062 }

```

(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.

```

4063 \cs_new_protected:Nn \__enumext_after_list_vii:
4064 {
4065   \__enumext_stop_mini_vii:
4066   \__enumext_after_stop_list_vii:

```

```

4067     \__enumext_check_ans_key_hook:
4068     \__enumext_vspace_below_vii:
4069     \bool_set_false:N \__enumext_starred_bool
4070     \__enumext_resume_save_counter:
4071 }

```

(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.

```

4072 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4073 {
4074     \bool_if:NT \__enumext_store_active_bool
4075     {
4076         \int_compare:nNtT { \__enumext_level_int } > { 0 }
4077         {
4078             \__enumext_store_level_open_vii:
4079         }
4080     }
4081 }
4082 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4083 {
4084     \bool_if:NT \__enumext_store_active_bool
4085     {
4086         \int_compare:nNtT { \__enumext_level_int } > { 0 }
4087         {
4088             \__enumext_store_level_close_vii:
4089         }
4090     }
4091 }

```

(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 \\_\_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.

```

4092 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4093 {
4094     \__enumext_stop_item_tmp_vii:
4095     \int_incr:N \__enumext_item_column_pos_vii_int
4096     \int_gincr:N \g__enumext_item_count_all_vii_int
4097     \__enumext_item_peek_args_vii:
4098 }

```

(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).

```

4099 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4100 {
4101     \peek_meaning:NTF (
4102     { \__enumext_joined_item_vii:w }
4103     { \__enumext_joined_item_vii:w (1) }
4104 }

```

(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.

```

4105 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4106 {
4107     \__enumext_starred_joined_item_vii:n {#1}
4108     \peek_meaning_remove:NTF *

```

```

4109     { \__enumext_starred_item_vii:w }
4110     { \__enumext_standar_item_vii:w }
4111 }

```

(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 ].

```

4112 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4113 {
4114     \bool_set_false:N \l__enumext_item_starred_vii_bool
4115     \peek_meaning:NTF [
4116     {
4117         \bool_set_eq:NN
4118         \l__enumext_wrap_label_vii_bool
4119         \l__enumext_wrap_label_opt_vii_bool
4120         \__enumext_start_item_vii:w
4121     }
4122     {
4123         \bool_set_true:N \l__enumext_wrap_label_vii_bool
4124         \legacy_if_set_true:n { @noitemarg }
4125         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4126     }
4127 }

```

(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

```

4128 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4129 {
4130     \bool_set_true:N \l__enumext_item_starred_vii_bool
4131     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4132     \peek_meaning:NTF [
4133     { \__enumext_starred_item_vii_aux_i:w }
4134     { \__enumext_starred_item_vii_aux_ii:w }
4135 }
4136 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4137 {
4138     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4139     \__enumext_starred_item_vii_aux_ii:w
4140 }
4141 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4142 {
4143     \peek_meaning:NTF [
4144     { \__enumext_starred_item_vii_aux_iii:w }
4145     {
4146         \dim_set_eq:NN
4147         \l__enumext_item_symbol_sep_vii_dim
4148         \l__enumext_labelsep_vii_dim
4149         \legacy_if_set_true:n { @noitemarg }
4150         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4151     }
4152 }
4153 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4154 {
4155     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4156     \legacy_if_set_true:n { @noitemarg }
4157     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4158 }

```

(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.

```

4159 \cs_new_protected_nopar:Npn \\_enumext_start_item_vii:w [#1]
4160 {
4161   \cs_set_eq:NN \\_enumext_stop_item_tmp_vii: \\_enumext_stop_item_vii:
4162   \legacy_if:nT { @noitemarg }
4163   {
4164     \legacy_if_set_false:n { @noitemarg }
4165     \legacy_if:nT { @nmbrrlist }
4166     {
4167       \bool_if:NT \\_enumext_hyperref_bool
4168       {
4169         \legacy_if_set_true:n { @hyper@item }
4170       }
4171       \refstepcounter{enumXvii}
4172       \bool_if:NT \\_enumext_check_answers_bool
4173       {
4174         \int_gincr:N \\_enumext_item_number_int
4175         \bool_set_true:N \\_enumext_item_number_bool
4176       }
4177     }
4178   }

```

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 `\\_enumext_footnotes_key_bool` is false, we will redefine the command `\footnote`, followed by printing the  $\langle symbol \rangle$  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.

```

4179 \group_begin:
4180   \lrbox{ \\_enumext_item_text_vii_box }
4181   \bool_if:NF \\_enumext_footnotes_key_bool
4182   {
4183     \\_enumext_renew_footnote:
4184   }
4185   \bool_if:NT \\_enumext_item_starred_vii_bool
4186   {
4187     \tl_if_blank:VT \\_enumext_item_symbol_aux_vii_tl
4188     {
4189       \tl_gset_eq:NN
4190       \\_enumext_item_symbol_aux_vii_tl \\_enumext_item_symbol_vii_tl
4191     }
4192     \mode_leave_vertical:
4193     \skip_horizontal:n { -\\_enumext_item_symbol_sep_vii_dim }
4194     \makebox[ 0pt ][ r ]{ \\_enumext_item_symbol_aux_vii_tl }
4195     \skip_horizontal:N \\_enumext_item_symbol_sep_vii_dim
4196     \tl_gclear:N \\_enumext_item_symbol_aux_vii_tl
4197   }
4198   \group_begin:
4199     \tl_use:N \\_enumext_label_font_style_vii_tl
4200     \bool_if:NTF \\_enumext_wrap_label_vii_bool
4201     {
4202       \makebox[ \\_enumext_labelwidth_vii_dim ][ \\_enumext_align_label_vii_str ]
4203       { \\_enumext_wrapper_label_vii:n {#1} }
4204     }
4205     {
4206       \makebox[ \\_enumext_labelwidth_vii_dim ][ \\_enumext_align_label_vii_str ]{ #1 }
4207     }
4208   \group_end:
4209   \skip_horizontal:N \\_enumext_labelsep_vii_dim
4210   \tl_use:N \\_enumext_after_list_args_vii_tl
4211   \\_enumext_minipage:w [ t ]{ \\_enumext_joined_width_vii_dim }
4212   \skip_set_eq:NN \parindent \\_enumext_listparindent_vii_dim

```

```

4213         \skip_set_eq:Nn \parskip \l__enumext_parsep_vii_skip
4214         \tl_use:N \l__enumext_fake_item_indent_vii_tl
4215     }

```

(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.

```

4216 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4217 {
4218     \__enumext_endminipage:
4219     \endlrbox
4220     \group_end:
4221     \box_set_wd:Nn \l__enumext_item_text_vii_box
4222     {
4223         \l__enumext_joined_width_vii_dim
4224         + \l__enumext_labelwidth_vii_dim
4225         + \l__enumext_labelsep_vii_dim
4226     }
4227     \int_set:Nn \hbadness { 10000 }
4228     \box_use_drop:N \l__enumext_item_text_vii_box
4229     \bool_if:NF \l__enumext_footnotes_key_bool
4230     {
4231         \__enumext_print_footnote:
4232     }
4233     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4234     {
4235         \par\noindent
4236         \int_zero:N \l__enumext_item_column_pos_vii_int
4237     }
4238     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4239 }

```

(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.

```

4240 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4241 {
4242     \int_compare:nNnTF
4243     {
4244         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4245     }
4246     =
4247     { 0 }
4248     {
4249         \par
4250         \vspace{ -\l__enumext_itemsep_vii_skip }
4251         \int_gzero:N \g__enumext_item_count_all_vii_int
4252     }
4253 }

```

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`.

```

4254 \__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.

```

4255 \NewDocumentEnvironment{keyans*}{ o }
4256 {
4257     \__enumext_safe_exec_viii:
4258     \__enumext_parse_keys_viii:n {#1}
4259     \__enumext_before_list_viii:
4260     \__enumext_start_list:nn { }
4261 }

```

```

4262         \__enumext_list_arg_two_viii:
4263         \__enumext_before_keys_exec_viii:
4264     }
4265     \__enumext_starred_columns_set_viii:
4266     \item[] \scan_stop:
4267     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4268     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4269 }
4270 {
4271     \__enumext_stop_item_tmp_viii:
4272     \__enumext_remove_extra_parsep_viii:
4273     \__enumext_check_starred_cmd:n { item }
4274     \__enumext_stop_list:
4275     \__enumext_after_list_viii:
4276 }

```

(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.

```

4277 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4278 {
4279     \int_incr:N \__enumext_keyans_level_h_int
4280     \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4281     {
4282         \msg_error:nn { enumext } { nested }
4283     }
4284     \__enumext_keyans_name_and_start:
4285     \bool_if:NT \__enumext_starred_bool
4286     {
4287         \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4288     }
4289     \bool_set_true:N \__enumext_starred_bool
4290     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4291     \bool_set_false:N \__enumext_store_active_bool
4292     \int_compare:nNnT { \__enumext_level_int } > { 1 }
4293     {
4294         \msg_error:nn { enumext } { keyans-wrong-level }
4295     }
4296 }

```

(End of definition for `\__enumext_safe_exec_viii:`)

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

```

4297 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4298 {
4299     \tl_if_novalue:nF {#1}
4300     {
4301         \keys_set:nn { enumext / keyans* } {#1}
4302     }
4303 }

```

(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`.

```

4304 \cs_new_protected:Nn \__enumext_before_list_viii:
4305 {
4306     \__enumext_vspace_above_viii:
4307     \__enumext_before_args_exec_viii:
4308     \__enumext_start_mini_viii:
4309 }

```

(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.

```

4310 \cs_new_protected:Nn \__enumext_after_list_viii:
4311 {
4312     \__enumext_stop_mini_viii:

```

```

4313     \__enumext_after_stop_list_viii:
4314     \__enumext_vspace_below_viii:
4315 }

```

(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 (`<number>`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `<label>` next to the `[<content>]` if it is present in the `<sequence>` and `<prop list>` defined by `save-ans` key for `\item*`, `\item* [<content>]`, `\item(<number>)*` and `\item(<number>)* [<content>]` 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`.

```

4316 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4317 {
4318     \__enumext_stop_item_tmp_viii:
4319     \int_incr:N \l__enumext_item_column_pos_viii_int
4320     \int_gincr:N \g__enumext_item_count_all_viii_int
4321     \__enumext_item_peek_args_viii:
4322 }

```

(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(<number>)`. Look for the argument “(”, if it is present we will call the function `\__enumext_joined_item_viii: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).

```

4323 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4324 {
4325     \peek_meaning:NTF (
4326         { \__enumext_joined_item_viii:w }
4327         { \__enumext_joined_item_viii:w (1) }
4328     }

```

(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`.

```

4329 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4330 {
4331     \__enumext_starred_joined_item_viii:n {#1}
4332     \peek_meaning_remove:NTF *
4333     { \__enumext_starred_item_viii:w }
4334     { \__enumext_standar_item_viii:w }
4335 }

```

(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 [<custom>]` by means of the function `\__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]`.

```

4336 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4337 {
4338     \bool_set_false:N \l__enumext_item_starred_viii_bool
4339     \peek_meaning:NTF [
4340     {
4341         \bool_set_eq:NN
4342         \l__enumext_wrap_label_viii_bool
4343         \l__enumext_wrap_label_opt_viii_bool

```



```

4344         \__enumext_start_item_viii:w
4345     }
4346 {
4347     \bool_set_true:N \__enumext_wrap_label_viii_bool
4348     \legacy_if_set_true:n { @noitemarg }
4349     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4350 }
4351 }

```

(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].

```

4352 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4353 {
4354     \bool_set_true:N \__enumext_item_starred_viii_bool
4355     \bool_set_true:N \__enumext_wrap_label_viii_bool
4356     \peek_meaning:NTF [
4357     { \__enumext_starred_item_viii_aux_i:w }
4358     { \__enumext_starred_item_viii_aux_ii:w }
4359 }

```

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.

```

4360 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4361 {
4362     \tl_clear:N \l__enumext_store_current_label_tl
4363     \tl_if_novalue:nF { #1 }
4364     {
4365         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4366         {
4367             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep }
4368             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4369         }
4370         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4371     }
4372     \__enumext_starred_item_viii_aux_ii:w
4373 }
4374 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4375 {
4376     \legacy_if_set_true:n { @noitemarg }
4377     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4378 }

```

(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 \langle label \rangle for \item\* followed by the [\langle content \rangle] for \item\*[\langle content \rangle] if present in the \langle sequence \rangle and \langle prop list \rangle set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```

4379 \cs_new_protected:Nn \__enumext_starred_item_exec:
4380 {
4381     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4382     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4383     \__enumext_keyans_store_ref:
4384     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4385     \__enumext_keyans_addto_seq_link:
4386     \int_gincr:N \g__enumext_check_starred_cmd_int
4387     \bool_if:NT \l__enumext_show_answer_bool
4388     {
4389         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4390     }
4391     \bool_if:NT \l__enumext_show_position_bool
4392     {
4393         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4394         {
4395             \group_begin:
4396             \exp_not:N \normalfont
4397             \exp_not:N \footnotesize [ \int_eval:n

```

```

4398         {
4399             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4400         }
4401     ]
4402     \group_end:
4403 }
4404 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4405 }
4406 }

```

(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.

```

4407 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4408 {
4409     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4410     \legacy_if:nT { @noitemarg }
4411     {
4412         \legacy_if_set_false:n { @noitemarg }
4413         \legacy_if:nT { @nmbrrlist }
4414         {
4415             \bool_if:NT \l__enumext_hyperref_bool
4416             {
4417                 \legacy_if_set_true:n { @hyper@item }
4418             }
4419             \refstepcounter{enumXviii}
4420         }
4421     }

```

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

```

4422 \group_begin:
4423 \lrbox{ \l__enumext_item_text_viii_box }
4424 \bool_if:NF \l__enumext_footnotes_key_bool
4425 {
4426     \__enumext_renew_footnote:
4427 }
4428 \bool_if:NT \l__enumext_item_starred_viii_bool
4429 {
4430     \__enumext_starred_item_exec:
4431 }
4432 \group_begin:
4433 \tl_use:N \l__enumext_label_font_style_viii_tl
4434 \bool_if:NTF \l__enumext_wrap_label_viii_bool
4435 {
4436     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4437     { \__enumext_wrapper_label_viii:n {#1} }
4438 }
4439 {
4440     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4441 }
4442 \group_end:
4443 \skip_horizontal:N \l__enumext_labelsep_viii_dim
4444 \tl_use:N \l__enumext_after_list_args_viii_tl
4445 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4446 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4447 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4448 \bool_if:NT \l__enumext_item_starred_viii_bool
4449 {
4450     \tl_use:N \l__enumext_fake_item_indent_viii_tl
4451     \__enumext_keyans_show_item_opt:
4452     \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4453 }
4454 {
4455     \tl_use:N \l__enumext_fake_item_indent_viii_tl
4456 }
4457 }

```

(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.

```

4458 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4459 {
4460     \__enumext_endminipage:
4461     \endlrbox
4462     \group_end:
4463     \box_set_wd:Nn \l__enumext_item_text_viii_box
4464     {
4465         \l__enumext_joined_width_viii_dim
4466         + \l__enumext_labelwidth_viii_dim
4467         + \l__enumext_labelsep_viii_dim
4468     }
4469     \int_set:Nn \hbadness { 10000 }
4470     \box_use_drop:N \l__enumext_item_text_viii_box
4471     \bool_if:NF \l__enumext_footnotes_key_bool
4472     {
4473         \__enumext_print_footnote:
4474     }
4475     \int_compare:nNnTF
4476     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4477     {
4478         \par\noindent
4479         \int_zero:N \l__enumext_item_column_pos_viii_int
4480     }
4481     { \hspace{ \l__enumext_columns_sep_viii_dim } }
4482 }

```

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

`\__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.

```

4483 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4484 {
4485     \int_compare:nNnT
4486     {
4487         \int_mod:nn
4488         { \g__enumext_item_count_all_viii_int }
4489         { \l__enumext_columns_viii_int }
4490     }
4491     =
4492     { 0 }
4493     {
4494         \par
4495         \vspace{ -\l__enumext_itemsep_viii_skip }
4496         \int_gzero:N \g__enumext_item_count_all_viii_int
4497     }
4498 }

```

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

## 12.45 The command `\getkeyans`

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form  $\{\langle store\ name\rangle:\langle position\rangle\}$ . Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from *prop list* defined by `save-ans` key.

```

4499 \NewDocumentCommand \getkeyans { m }
4500 {
4501     \exp_args:Ne \__enumext_getkeyans_aux:n
4502     { \tl_to_str:e { \text_expand:n {#1} } }
4503 }

```

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

`\__enumext_getkeyans_aux:n` The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the *argument* using “.”. If “.” is omitted it will return an error.

```

4504 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4505 {
4506     \str_if_in:nnTF {#1} { : }
4507     {
4508         \use:e

```

```

4509         {
4510             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4511             { {##1} {##2} }
4512         }
4513         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4514     }
4515     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4516 }

```

(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*.

```

4517 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4518 {
4519     \prop_if_exist:cTF { g__enumext_#1_prop }
4520     {
4521         \prop_item:cn { g__enumext_#1_prop }{#2}
4522     }
4523     {
4524         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4525     }
4526 }

```

(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⟩]*.

```

4527 \keys_define:nn { enumext / print }
4528 {
4529     print* .code:n      = \keys_precompile:neN { enumext / enumext* }
4530               { \__enumext_filter_save_key:n {#1} }
4531               \l__enumext_print_keyans_starred_tl, % starred cmd
4532     print* .initial:n   = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4533     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4534               { \__enumext_filter_save_key:n {#1} }
4535               \l__enumext_print_keyans_i_tl,
4536     print-1 .initial:n  = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4537     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4538               { \__enumext_filter_save_key:n {#1} }
4539               \l__enumext_print_keyans_ii_tl,
4540     print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4541     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4542               { \__enumext_filter_save_key:n {#1} }
4543               \l__enumext_print_keyans_iii_tl,
4544     print-3 .initial:n  = { nosep, label=\roman*, first=\small, font=\small },
4545     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4546               { \__enumext_filter_save_key:n {#1} }
4547               \l__enumext_print_keyans_iv_tl,
4548     print-4 .initial:n  = { nosep, label=\Alph*, first=\small, font=\small },
4549     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4550               { \__enumext_filter_save_key:n {#1} }
4551               \l__enumext_print_keyans_vii_tl, % starred nested
4552     print-* .initial:n  = { nosep, label=\arabic*, first=\small, font=\small },
4553 }

```

- 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`.

```

4554 \NewDocumentCommand \printkeyans { s O{} m }
4555 {
4556   \group_begin:
4557     \tl_use:N \__enumext_print_keyans_i_tl
4558     \tl_use:N \__enumext_print_keyans_ii_tl
4559     \tl_use:N \__enumext_print_keyans_iii_tl
4560     \tl_use:N \__enumext_print_keyans_iv_tl
4561     \tl_use:N \__enumext_print_keyans_vii_tl
4562     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4563   \group_end:
4564 }

```

(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.

```

4565 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4566 {
4567   \seq_if_exist:cTF { g__enumext_#3_seq }
4568   {
4569     \seq_if_empty:cF { g__enumext_#3_seq }
4570     {
4571       %%\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 `\__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⟩*.

```

4572       \bool_if:nTF {#1}
4573       {
4574         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4575         {
4576           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4577         }
4578         {
4579           \tl_use:N \__enumext_print_keyans_starred_tl
4580           \begin{enumext*}[#2]
4581             \keys_set:nn { enumext / level-1 }{ base-fix }
4582             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4583             \end{enumext*}
4584           }
4585         }

```

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⟩*.

```

4586         {
4587           \begin{enumext}[#2]
4588             \keys_set:nn { enumext / enumext* }{ base-fix }
4589             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4590             \end{enumext}
4591           }
4592         }
4593       }
4594     {
4595       \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4596     }
4597   }

```

(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`.

```

4598 \cs_new:Npn \__enumext_filter_first_level:n #1

```

```

4599 {
4600   \use:e
4601   {
4602     \keyval_parse:NNn
4603     \__enumext_filter_first_level_key:n
4604     \__enumext_filter_first_level_pair:nn {#1}
4605   }
4606 }

```

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*`.

```

4607 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4608 {
4609   \str_case:nnF {#1}
4610   {
4611     { resume } {}
4612     { resume* } {}
4613   }
4614   { , { \exp_not:n {#1} } }
4615 }

```

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.

```

4616 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4617 {
4618   \str_case:nnF {#1}
4619   {
4620     { series } {}
4621     { resume } {}
4622     { save-ans } {}
4623   }
4624   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
4625 }

```

(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`.

```

4626 \keys_define:nn { enumext / meta-families }
4627 {
4628   enumext-1 .code:n =
4629   {
4630     \keys_set:ne { enumext / level-1 }
4631     {
4632       \__enumext_filter_first_level:n {#1}
4633     }
4634   } ,
4635   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4636   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4637   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4638   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4639   enumext* .code:n =
4640   {
4641     \keys_set:ne { enumext / enumext* }
4642     {
4643       \__enumext_filter_first_level:n {#1}
4644     }
4645   } ,
4646   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4647   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4648   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4649   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4650   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4651   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4652   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4653   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4654 }

```

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

```

4655 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4656 {
4657   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4658   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4659 }

```

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

```

4660 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4661 {
4662   \seq_clear:N \l__enumext_setkey_tmpa_seq
4663   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4664   \int_set:Nn \l__enumext_setkey_tmpa_int
4665   {
4666     \seq_count:N \l__enumext_setkey_tmpb_seq
4667   }
4668   \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4669   {
4670     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4671     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4672     \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4673     {
4674       \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4675     }
4676   }
4677   {
4678     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4679   }
4680   \seq_if_empty:NNTF \l__enumext_setkey_tmpa_seq
4681   { \seq_map_inline:Nn \c__enumext_all_families_seq }
4682   { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4683   {
4684     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4685   }
4686 }

```

(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.

```

4687 \cs_new_protected:Npn \__enumext_set_parse:n #1
4688 {
4689   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4690   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4691   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4692   \tl_if_empty:NNTF \l__enumext_setkey_tmpb_tl
4693   {
4694     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4695     { \tl_trim_spaces:n {#1} }
4696   }
4697   { \__enumext_set_error:nn {#1} { } }
4698 }
4699 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4700 { \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
4701 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
4702 {
4703   {enumext,1} = level-1,
4704   {enumext,2} = level-2,
4705   {enumext,3} = level-3,
4706   {enumext,4} = level-4,
4707   {enumext*} = enumext*
4708 }

```

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

```

4709 \NewDocumentCommand \setenumextmeta { s 0{enumext,1} m +m }
4710 {
4711   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
4712   { \msg_error:nn { enumext } { prohibited-unknown } }
4713   {
4714     \bool_if:nTF {#1}

```



```

4715         {
4716             \int_step_inline:nn { 4 }
4717             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
4718             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
4719         }
4720     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
4721 }
4722 }

```

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

```

4723 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
4724 {
4725     \tl_set:Nn \l__enumext_meta_path_tl {#1}
4726     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
4727     \prop_get:NVNTF
4728     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
4729     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
4730     {
4731         \msg_error:nnn { enumext } { unknown-set } {#1}
4732         \use_none:nn
4733     }
4734 }
4735 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
4736 {
4737     \bool_lazy_or:nnTF
4738     { \keys_if_exist:p:nn { enumext / #1 } {#2} }
4739     { \keys_if_exist:p:nn { enumext / enumext* } {#2} }
4740     { \msg_error:nnn { enumext } { already-defined } {#2} }
4741     {
4742         \keys_define:nn { enumext / #1 }
4743         {
4744             #2 .meta:n = {#3},
4745             #2 .value_forbidden:n = true
4746         }
4747     }
4748 }
4749 \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.

```

4750 \keys_define:nn { enumext / foreach }
4751 {
4752     before .tl_set:N = \l__enumext_foreach_before_tl,
4753     before .value_required:n = true,
4754     after .tl_set:N = \l__enumext_foreach_after_tl,
4755     after .value_required:n = true,
4756     start .int_set:N = \l__enumext_foreach_start_int,
4757     start .value_required:n = true,
4758     stop .int_set:N = \l__enumext_foreach_stop_int,
4759     stop .value_required:n = true,
4760     step .int_set:N = \l__enumext_foreach_step_int,
4761     step .value_required:n = true,
4762     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
4763     wrapper .value_required:n = true,
4764     sep .tl_set:N = \l__enumext_foreach_sep_tl,
4765     sep .value_required:n = true,
4766     unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
4767 }
4768 \keys_precompile:nnN { enumext / foreach }
4769 {
4770     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
4771 }
4772 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown  $\langle keys \rangle$ .

```

4773 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
4774 {
4775   \tl_if_blank:nTF {#2}
4776   {
4777     \msg_error:nnn { enumext } { for-key-unknown } {#1}
4778   }
4779   {
4780     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
4781   }
4782 }
4783 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
4784 {
4785   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
4786 }

```

We create the command.

```

4787 \NewDocumentCommand \foreachkeyans { +0{} m }
4788 {
4789   \__enumext_foreach_keyans:nn {#1} {#2}
4790 }

```

Finally the internal functions `\__enumext_foreach_keyans:nn` and `\__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

4791 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
4792 {
4793   \tl_use:N \g__enumext_foreach_default_keys_tl
4794   \keys_set:nn { enumext / foreach } {#1}
4795   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
4796   \prop_if_exist:cF { g__enumext_#2_prop }
4797   {
4798     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
4799   }
4800   \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
4801   {
4802     \int_set:Nn \l__enumext_foreach_stop_int
4803     { \prop_count:c { g__enumext_#2_prop } }
4804   }
4805   \seq_clear:N \l__enumext_foreach_print_seq
4806   \int_step_function:nnnN
4807   { \l__enumext_foreach_start_int }
4808   { \l__enumext_foreach_step_int }
4809   { \l__enumext_foreach_stop_int }
4810   \__enumext_foreach_add_body:n
4811   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
4812 }
4813 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
4814 {
4815   \seq_put_right:Ne \l__enumext_foreach_print_seq
4816   {
4817     \exp_not:V \l__enumext_foreach_before_tl
4818     \__enumext_foreach_wrapper:n
4819     {
4820       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
4821     }
4822     \exp_not:V \l__enumext_foreach_after_tl
4823   }
4824 }

```

(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.

```

4825 \msg_new:nnn { enumext } { package-load }
4826 {
4827   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4828 }
4829 \msg_new:nnn { enumext } { package-not-load }
4830 {
4831   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4832 }

```

```

4833 \msg_new:nnn { enumext } { package-load-foot }
4834 {
4835   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4836 }

```

Message used in the creation of counters by **enumext** package.

```

4837 \msg_new:nnn { enumext } { counters }
4838 {
4839   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4840   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4841 }

```

Message used by **align** and **mark-pos** keys.

```

4842 \msg_new:nnn { enumext } { unknown-choice }
4843 {
4844   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4845 }

```

Message used by reserved **anskey\*** environment by **enumext** package.

```

4846 \msg_new:nnnn { enumext } { anskey-env-error }
4847 {
4848   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4849   'enumext' ~ package, ~ It~ is~ already~ defined.
4850 }
4851 {
4852   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4853   for ~ the ~ 'save-ans' ~ key.\\
4854 }

```

Message used in the creation of *(prop list)* by **enumext** package.

```

4855 \msg_new:nnn { enumext } { store-prop }
4856 {
4857   * ~ Package ~ enumext: ~ Creating ~
4858   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4859 }
4860 \msg_new:nnn { enumext } { store-seq }
4861 {
4862   * ~ Package ~ enumext: ~ Creating ~
4863   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4864 }
4865 \msg_new:nnn { enumext } { store-int }
4866 {
4867   * ~ Package ~ enumext: ~ Creating ~
4868   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4869 }
4870 \msg_new:nnn { enumext } { prop-seq-int-hook }
4871 {
4872   * ~ Package ~ enumext: ~ Elements ~ in ~
4873   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4874   * ~ Package ~ enumext: ~ Elements ~ in ~
4875   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4876   * ~ Package ~ enumext: ~ Value ~ off ~
4877   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4878 }
4879 \msg_new:nnn { enumext } { item-answer-hook }
4880 {
4881   * ~ Package ~ enumext: ~ Value ~ off ~
4882   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4883   * ~ Package ~ enumext: ~ Value ~ off ~
4884   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4885   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4886 }

```

Message used by *[key = val]* system and **\setenumext** command.

```

4887 \msg_new:nnn { enumext } { invalid-key }
4888 {
4889   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4890 }
4891 \msg_new:nnn { enumext } { unknown-key-family }
4892 {
4893   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4894 }

```

Messages used in length calculation.

```

4895 \msg_new:nnn { enumext } { width-negative }
4896 {
4897   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4898   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4899 }
4900 \msg_new:nnn { enumext } { width-zero }
4901 {
4902   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4903   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4904 }

```

Messages used by `show-length` key in `enumext`.

```

4905 \msg_new:nnn { enumext } { list-lengths }
4906 {
4907   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4908   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4909   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4910   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4911   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4912   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4913   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4914   \__enumext_show_length:nnn { skip } { topsep } {#1}
4915   \__enumext_show_length:nnn { skip } { parsep } {#1}
4916   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4917   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4918   ****
4919 }

```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```

4920 \msg_new:nnn { enumext } { list-lengths-not-nested }
4921 {
4922   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4923   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4924   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4925   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4926   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4927   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4928   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4929   \__enumext_show_length:nnn { skip } { topsep } {#1}
4930   \__enumext_show_length:nnn { skip } { parsep } {#1}
4931   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4932   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4933   ****
4934 }

```

Messages used by `ref` key.

```

4935 \msg_new:nnn { enumext } { key-ref-empty }
4936 {
4937   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4938 }

```

Messages used by `save-ans` key.

```

4939 \msg_new:nnn { enumext } { save-ans-empty }
4940 {
4941   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4942 }
4943 \msg_new:nnn { enumext } { save-ans-log }
4944 {
4945   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4946 }
4947 \msg_new:nnn { enumext } { save-ans-log-hook }
4948 {
4949   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4950 }
4951 \msg_new:nnn { enumext } { save-ans-hook }
4952 {
4953   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4954 }

```

Messages used by the internal system to check answer used by `check-ans` key.

```

4955 \msg_new:nnn { enumext } { need-save-ans }
4956 {

```

```

4957     Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
4958 }
4959 \msg_new:nnn { enumext } { items-same-answer }
4960 {
4961     *****\
4962     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4963     for ~ \c_left_brace_str #2 \c_right_brace_str\
4964     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4965     'OK', ~ all ~ items ~ with ~ answer.\
4966     *****
4967 }
4968 \msg_new:nnn { enumext } { item-greater-answer }
4969 {
4970     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4971     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
4972     Items ~ > ~ Answers.
4973 }
4974 \msg_new:nnn { enumext } { item-less-answer }
4975 {
4976     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4977     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
4978     Items ~ < ~ Answers.
4979 }

```

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```

4980 \msg_new:nnn { enumext } { missing-starred }
4981 {
4982     Missing ~ '\c_backslash_str #1*' ~ #2.
4983 }
4984 \msg_new:nnn { enumext } { many-starred }
4985 {
4986     Many ~ '\c_backslash_str #1*' ~ #2.
4987 }

```

Messages used by `\printkeyans*` command.

```

4988 \msg_new:nnn { enumext } { print-starred }
4989 {
4990     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4991     #2 ~ environment ~ \msg_line_context:.
4992 }

```

Message for the nesting depth of the environment `enumext`.

```

4993 \msg_new:nnn { enumext } { list-too-deep }
4994 {
4995     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
4996     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4997 }

```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```

4998 \msg_new:nnn { enumext } { anskey-unnumber-item }
4999 {
5000     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5001 }
5002 \msg_new:nnn { enumext } { anskey-already-stored }
5003 {
5004     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5005 }
5006 \msg_new:nnn { enumext } { anskey-empty-arg }
5007 {
5008     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5009 }
5010 \msg_new:nnn { enumext } { anskey-wrong-place }
5011 {
5012     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
5013     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5014 }
5015 \msg_new:nnn { enumext } { anskey-nested }
5016 {
5017     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5018 }
5019 \msg_new:nnn { enumext } { anskey-math-mode }
5020 {
5021     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.

```

```

5022     }
5023     \msg_new:nnn { enumext } { anskey-env-wrong }
5024     {
5025         The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5026     }
5027     \msg_new:nnn { enumext } { ansPIC-wrong-place }
5028     {
5029         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5030         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5031     }
5032     \msg_new:nnn { enumext } { command-wrong-place }
5033     {
5034         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5035         '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5036     }
5037     \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5038     {
5039         The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5040         'anskey*' ~ and ~ is ~ being ~ ignored.
5041     }
5042     {
5043         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5044         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5045     }
5046     \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5047     {
5048         The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5049         'anskey*' ~ and ~ is ~ being ~ ignored.
5050     }
5051     {
5052         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5053         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5054     }
5055     \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5056     { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5057     {
5058         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5059         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5060     }
5061     \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5062     { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5063     {
5064         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5065         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5066     }

```

Messages used by `keyans`, `keyans*` and `keyansPIC` environment.

```

5067     \msg_new:nnn { enumext } { keyans-nested }
5068     {
5069         The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5070     }
5071     \msg_new:nnn { enumext } { keyans-wrong-level }
5072     {
5073         Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5074         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5075     }
5076     \msg_new:nnn { enumext } { wrong-place }
5077     {
5078         Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
5079         '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5080     }
5081     \msg_new:nnn { enumext } { keyansPIC-nested }
5082     {
5083         The ~ environment ~ 'keyansPIC' ~ can't ~ be ~ nested~ \msg_line_context:~.
5084     }
5085     \msg_new:nnn { enumext } { keyansPIC-wrong-level }
5086     {
5087         Wrong ~ level ~ position ~ for ~ 'keyansPIC' ~ \msg_line_context:~ \\
5088         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5089     }
5090     \msg_new:nnn { enumext } { keyansPIC-item-cmd }
5091     {

```

```

5092     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5093 }
5094 \msg_new:nnnn { enumext } { keyans-unknown-key }
5095 {
5096     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5097     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5098 }
5099 {
5100     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5101     ~ have ~ a ~ key ~ called ~'#1'.\\
5102     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5103 }
5104 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5105 {
5106     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5107     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5108 }
5109 {
5110     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5111     ~ have ~ a ~ key ~ called ~'#1'.\\
5112     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5113 }

```

Message used by unknown *⟨keys⟩* in *enumext\**. environment.

```

5114 \msg_new:nnnn { enumext } { starred-unknown-key }
5115 {
5116     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5117     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5118 }
5119 {
5120     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5121     ~ have ~ a ~ key ~ called ~'#1'.\\
5122     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5123 }
5124 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5125 {
5126     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5127     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5128 }
5129 {
5130     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5131     ~ have ~ a ~ key ~ called ~'#1'.\\
5132     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5133 }

```

Message used by unknown *⟨keys⟩* in *enumext* environment.

```

5134 \msg_new:nnnn { enumext } { standar-unknown-key }
5135 {
5136     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5137     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5138 }
5139 {
5140     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5141     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5142     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5143 }
5144 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5145 {
5146     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5147     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5148 }
5149 {
5150     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5151     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5152     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5153 }

```

Message used by unknown *⟨keys⟩* in *\foreachkeyans*.

```

5154 \msg_new:nnnn { enumext } { for-key-unknown }
5155 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5156 {
5157     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\

```



```

5158     Check~that~you~have~spelled~the~key~name~correctly.
5159 }
5160 \msg_new:nnnn { enumext } { for-key-value-unknown }
5161 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5162 {
5163     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5164     Check~that~you~have~spelled~the~key~name~correctly.
5165 }

```

Messages used by `\getkeyans` command.

```

5166 \msg_new:nnn { enumext } { undefined-storage-anskey }
5167 {
5168     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5169 }

```

Messages used by `\miniright` command.

```

5170 \msg_new:nnn { enumext } { missing-miniright }
5171 {
5172     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5173     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5174 }
5175 \msg_new:nnn { enumext } { wrong-miniright-place }
5176 {
5177     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5178     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5179 }
5180 \msg_new:nnn { enumext } { wrong-miniright-use }
5181 {
5182     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5183     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5184 }
5185 \msg_new:nnn { enumext } { wrong-miniright-starred }
5186 {
5187     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5188 }
5189 \msg_new:nnn { enumext } { many-miniright-used }
5190 {
5191     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5192 }

```

Messages used by `\setenumextmeta` command.

```

5193 \msg_new:nnn { enumext } { unknown-set }
5194 {
5195     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5196 }
5197 \msg_new:nnn { enumext } { already-defined }
5198 {
5199     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5200 }
5201 \msg_new:nnn { enumext } { prohibited-unknown }
5202 {
5203     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5204 }

```

Messages used by `enumext*` and `keyans*` environments.

```

5205 \msg_new:nnn { enumext } { nested }
5206 {
5207     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5208 }
5209 \msg_new:nnn { enumext } { nested-horizontal }
5210 {
5211     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ 
5212 }
5213 \msg_new:nnn { enumext } { item-joined }
5214 {
5215     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
5216 }
5217 \msg_new:nnn { enumext } { item-joined-columns }
5218 {
5219     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5220 }

```

12.51 Finish package

Finish package implementation.

```
5221 \file_input_stop:  
5222 </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

\\*

219

\+

211

\-

211

\\

227, 2607, 3602, 4839, 4848, 4853, 4873, 4875, 4882, 4884, 4897, 4902, 4907, 4922, 4961, 4963, 4965, 4970, 4971, 4976, 4977, 4995, 5012, 5029, 5034, 5043, 5052, 5058, 5064, 5073, 5078, 5087, 5101, 5111, 5121, 5131, 5141, 5151, 5157, 5163, 5172, 5177, 5182

A

above

1422

above\*

1422

\addvspace

1121, 1149, 1220, 1268, 1331, 1337, 1380, 1408, 3419, 3555, 3571, 3929, 3943, 3984, 3998

after

960

align

509

\Alph

36, 41

\Alph

461, 576, 621, 689, 4548

\alph

36, 41

\alph

462, 574, 4540

\anskey

12, 72, 74, 2425

anskey\*

13, 2535

\anspic

15, 98, 3580

\anspic\*

66

\arabic

30, 36

\arabic

460, 573, 620, 4532, 4536, 4552

B

base-fix

819

\baselineskip

50

\baselineskip

836, 847

before

960

before\*

960

below

1422

below\*

1422

bool commands:

\bool\_gset\_false:N

332, 333, 334, 2711, 2713, 3945, 3949, 4000

\bool\_gset\_true:N

240, 250, 1063, 1915, 1921, 3921, 3946, 3976, 4001

\bool\_if:NTF

400, 412, 429, 1353, 1444, 1458, 1471, 1482, 1493, 1504, 1515, 1526, 1575, 1592, 1597, 1605, 1632, 1670, 1675, 1682, 1686, 1708, 1713, 1721, 1728, 1759, 1767, 1860, 2058, 2068, 2147, 2171, 2178, 2202, 2300, 2322, 2362, 2375, 2379, 2429, 2448, 2472, 2526, 2537, 2626, 2663, 2727, 2760, 2775, 2850, 2861, 2865, 2884, 2897, 2939, 2973, 3008, 3142, 3204, 3214, 3246, 3251, 3354, 3402, 3417, 3425, 3484, 3539, 3553, 3561, 3582, 3917, 3926, 3930, 3972, 3981, 3985, 4074, 4084, 4167, 4172, 4181, 4185, 4200, 4229, 4285, 4387, 4391, 4415, 4424, 4428, 4434, 4448, 4471

\bool\_if:nTF

1381, 1409, 2995, 3124, 3162, 3603, 4572, 4714

\bool\_if\_p:N

259, 274, 832, 833, 843, 844, 1739, 1740, 1748, 1749, 1873, 1899, 1912, 1913, 1918, 1919, 2235, 2245, 2257, 2272, 2273, 2307, 2348, 2349, 2649, 2837, 2838, 2875, 2876, 3327, 3329, 3340, 3610, 3611

\bool\_lazy\_all:nTF

257, 272, 1871, 1897, 2233, 2242, 2255, 2270, 3325, 3338

\bool\_lazy\_and:nnTF

236, 246, 831, 842, 1346, 1738, 1747, 1911, 1917, 2306, 2313, 2347, 2490, 2502, 2648, 2654, 2836

\bool\_lazy\_or:nnTF

1800, 1807, 2874, 3609, 4737

\bool\_new:N

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

\bool\_not\_p:n

237, 247, 2244, 2308, 2314, 2650, 2655, 3328, 3341

\bool\_set\_eq:NN

2948, 3104, 4117, 4341

\bool\_set\_false:N

409, 853, 1845, 1846, 1878, 1883, 1887, 1891, 1904, 2590, 3302, 3442, 3492, 3576, 3641, 3659, 4042, 4069, 4114, 4291, 4338

\bool\_set\_true:N

264, 265, 279, 280, 391, 395, 502, 868, 1428, 1433, 1695, 1817, 1818, 2090, 2098, 2591, 2942, 2944, 2976, 2978, 3100, 3112, 3226, 3301, 3334, 3347, 3373, 3489, 3516, 3906, 3961, 4041, 4123, 4130, 4131, 4175, 4289, 4347, 4354, 4355

box commands:

\box\_dp:N

1277, 1278, 1281, 1288, 1301, 1309, 1315, 1323, 3671

\box\_ht:N

1183, 1192, 1201, 1220, 1244, 1268

\box\_new:N

70, 174, 180

\box\_set\_wd:Nn

4221, 4463

\box\_use\_drop:N

3941, 3996, 4228, 4470

\box\_wd:N

468

C

\c

219, 220, 726, 728, 740, 742

\catcode

2607

\cB

220

\cE

220

\centering

1383, 1411, 3697, 3934, 3989

check-ans

1837

Document class:

article

42

clist commands:

\clist\_const:Nn

186

\clist\_map\_function:nN

3684

\clist\_map\_inline:Nn

508, 774, 959, 974, 1055, 1438

\clist\_map\_inline:nn

49, 60, 78, 85, 97, 109, 138, 161, 185, 536, 556, 828, 873, 894, 1069, 1544, 1784, 1851, 2037, 2055, 2087, 2230, 2769, 3029, 3041, 3081, 3191, 3194, 3221, 3233, 3236, 3256, 4690

\columnbreak

73

\columnbreak

2310

columns

1039

columns-sep

1039

\columnsep

94

\columnsep

3397, 3537

\columnseprule

94

\columnseprule

3400, 3538

Commands provide by enumext:

\anskey

28, 63, 68–72, 74, 75, 81, 83, 93, 108, 116, 118, 125

\anspic\*

28, 29, 66, 69, 81, 82, 98, 99, 101, 116, 118

\anspic

70, 98–100, 125

\foreachkeyans

121, 127

\getkeyans

69, 116, 128

\item\*

28, 29, 66, 69, 70, 81, 82, 85, 88, 109, 114, 116, 118

\item

85, 88, 103, 108–110, 113

©2024 by Pablo González L

130 / 144

\miniright . . . . .	27, 47, 53, 54, 94, 95, 128
\printkeyans* . . . . .	117
\printkeyans . . . . .	28, 70, 117
\setenumextmeta . . . . .	120, 128
\setenumext . . . . .	28, 117, 119, 120, 123
Counters defined by <b>enumext</b> :	
enumXiii . . . . .	26, 36
enumXii . . . . .	26, 36
enumXiv . . . . .	26, 36
enumXi . . . . .	26, 36
enumXviii . . . . .	26, 36
enumXvii . . . . .	26, 36, 110
enumXvi . . . . .	26, 36
enumXv . . . . .	26, 36
cs commands:	
\cs_generate_variant:Nn . . . . .	191, 192, 470, 486, 732, 748, 2139, 2144, 2220, 2543, 3181, 3686, 4749
\cs_if_exist:NTF . . . . .	440
\cs_if_free:NTF . . . . .	2494, 2506
\cs_new:Nn . . . . .	205
\cs_new:Npn . . . . .	223, 1545, 1554, 1562, 2102, 2111, 2119, 4598, 4607, 4616
\cs_new_eq:NN . . . . .	359, 360, 361, 365, 366, 414, 415, 418, 419
\cs_new_protected:Nn . . . . .	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, 1070, 1082, 1106, 1123, 1134, 1151, 1176, 1209, 1222, 1257, 1270, 1292, 1327, 1333, 1439, 1453, 1467, 1478, 1489, 1500, 1511, 1522, 1603, 1706, 1719, 1736, 1757, 1785, 1790, 1815, 1856, 1866, 1909, 1924, 1931, 1940, 1945, 1950, 1955, 1964, 1969, 1974, 2145, 2169, 2176, 2200, 2207, 2221, 2446, 2465, 2481, 2544, 2580, 2611, 2646, 2688, 2709, 2717, 2758, 2773, 2801, 2834, 2870, 2882, 2895, 2981, 2991, 3002, 3120, 3136, 3277, 3294, 3323, 3352, 3359, 3381, 3411, 3423, 3465, 3482, 3506, 3523, 3548, 3559, 3599, 3643, 3657, 3682, 3687, 3703, 3707, 3726, 3736, 3767, 3896, 3915, 3951, 3970, 4028, 4056, 4063, 4072, 4082, 4099, 4240, 4277, 4304, 4310, 4323, 4379, 4483
\cs_new_protected:Npn . . . . .	193, 197, 201, 422, 438, 455, 465, 471, 577, 622, 694, 719, 733, 1363, 1394, 1571, 1590, 1660, 1693, 1795, 1979, 2056, 2066, 2088, 2096, 2131, 2140, 2296, 2359, 2373, 2411, 2415, 2535, 2566, 2570, 2601, 2737, 2811, 2855, 2935, 2954, 3042, 3046, 3060, 3064, 3082, 3086, 3096, 3108, 3150, 3184, 3224, 3305, 3502, 3652, 3798, 3847, 4045, 4105, 4112, 4128, 4136, 4141, 4153, 4297, 4329, 4336, 4352, 4360, 4374, 4504, 4517, 4565, 4687, 4699, 4723, 4735, 4773, 4783, 4791, 4813
\cs_new_protected_nopar:Nn . . . . .	4092, 4216, 4316, 4458
\cs_new_protected_nopar:Npn . . . . .	4159, 4407
\cs_set:Npn . . . . .	2231, 2268, 4510
\cs_set_eq:NN . . . . .	4018, 4019, 4161, 4267, 4268, 4409
\cs_set_protected:Nn . . . . .	899, 915, 927, 939
\cs_set_protected:Npn . . . . .	45, 54, 71, 79, 92, 98, 131, 157, 165, 487, 509, 541, 557, 604, 749, 775, 819, 855, 878, 951, 960, 1039, 1056, 1422, 1533, 1776, 1837, 1996, 2038, 2074, 2223, 2762, 3018, 3034, 3074, 3182, 3222
\cs_to_str:N . . . . .	457, 480
\cs_undefine:N . . . . .	2483, 2484, 2485, 2486
<b>D</b>	
\d . . . . .	211
<b>E</b>	
\DeclareDocumentEnvironment . . . . .	372
dim commands:	
\dim_abs:n . . . . .	3155, 3160
\dim_add:Nn . . . . .	3662, 3761, 3792
\dim_compare:nNnTF . . . . .	901, 917, 929, 941, 1365, 1396, 3152, 3157, 3163, 3169, 3171, 3173, 3364, 3386, 3510, 3527, 3654, 3738, 3754, 3769, 3785, 3898, 3953
\dim_compare:nTF . . . . .	2332, 2676, 3283, 3471
\dim_gset_eq:NN . . . . .	3907, 3962
\dim_gzero:N . . . . .	2715, 3948, 4003
\dim_new:N . . . . .	67, 74, 75, 76, 94, 140, 173, 175, 176, 182
\dim_set:Nn . . . . .	468, 869, 2971, 3155, 3160, 3162, 3165, 3166, 3170, 3172, 3175, 3176, 3178, 3279, 3367, 3389, 3467, 3512, 3529, 3689, 3740, 3747, 3771, 3778, 3833, 3882, 3900, 3955, 4155
\dim_set_eq:NN . . . . .	564, 611, 682, 686, 2886, 2887, 2899, 2900, 2966, 3193, 3235, 3397, 3537, 3840, 3843, 3844, 3889, 3892, 3893, 4146
\dim_sub:Nn . . . . .	3288, 3476, 3756, 3787
\dim_use:N . . . . .	902, 910, 1366, 1377, 2210, 2213, 2218, 2986, 2988, 3285, 3290, 3365, 3370, 3371, 3377, 3387, 3391, 3392, 3394
\dim_zero:N . . . . .	3227, 3400, 3538, 3663, 3664, 3665
\dim_zero_new:N . . . . .	437
\c_zero_dim . . . . .	904, 918, 930, 942, 1366, 1396, 2334, 2678, 3152, 3157, 3163, 3170, 3285, 3365, 3387, 3473, 3510, 3527, 3738, 3754, 3769, 3785, 3898, 3953
\dimeval . . . . .	2003
<b>E</b>	
\end . . . . .	1374, 1403, 2173, 2204, 3416, 3434, 3552, 3570, 3919, 3942, 3974, 3997, 4574, 4583, 4590
\endgroup . . . . .	2607
\endlist . . . . .	360
\endlrbox . . . . .	4219, 4461
\endminipage . . . . .	366
enumext . . . . .	5, <u>3257</u>
enumext internal commands:	
\l__enumext__ref_the_count_tl . . . . .	38
\l__enumext__resume_name_tl . . . . .	59
\__enumext_add_meta_key:nnn . . . . .	121, <u>4701</u> , 4717, 4718, 4720, 4723
\__enumext_add_pre_parsep: . . . . .	48, 1080, <u>1082</u> , 1082
\__enumext_after_args_exec: . . . . .	46, <u>975</u> , 987, 3270
\__enumext_after_args_exec_v: . . . . .	991, 1003, 3458
\__enumext_after_args_exec_vii: . . . . .	<u>1007</u> , 1031
\__enumext_after_args_exec_viii: . . . . .	1035
\__enumext_after_env:nn . . . . .	78, 79, 81, 96, 105, 111, 197, 197, 2621, 3445, 3924, 3979, 4254
\__enumext_after_hyperref: . . . . .	34, 384, <u>386</u> , 386
\__enumext_after_list: . . . . .	95, 112, 3275, <u>3423</u> , 3423
\l__enumext_after_list_args_v_tl . . . . .	1005
\l__enumext_after_list_args_vii_tl . . . . .	1033, 4210
\l__enumext_after_list_args_viii_tl . . . . .	1037, 4444
\__enumext_after_list_v: . . . . .	3463, <u>3506</u> , 3559
\__enumext_after_list_vii: . . . . .	107, 4026, <u>4063</u> , 4063
\__enumext_after_list_viii: . . . . .	4275, <u>4310</u> , 4310
\__enumext_after_stop_list: . . . . .	46, 95, <u>975</u> , 983, 3439
\__enumext_after_stop_list_v: . . . . .	991, 999, 3577
\l__enumext_after_stop_list_v_tl . . . . .	1001
\__enumext_after_stop_list_vii: . . . . .	107, <u>1007</u> , 1023, 4066
\l__enumext_after_stop_list_vii_tl . . . . .	1025

```

\__enumext_after_stop_list_viii: . 1027, 4313
\l__enumext_after_stop_list_viii_tl ... 1029
\l__enumext_align_label_vii_str .. 4202, 4206
\l__enumext_align_label_viii_str . 4436, 4440
\l__enumext_align_label_X_str ..... 165
\c__enumext_all_envs_clist .. 186, 508, 774, 959,
    974, 1055, 1438
\c__enumext_all_families_seq .. 119, 4655, 4681
\l__enumext_anskey_env_bool 31, 77, 34, 265, 280,
    2537
\__enumext_anskey_env_clean_vars: . 80, 2642,
    2646, 2709
\__enumext_anskey_env_define_keys: 77, 2535,
    2544, 2615
\__enumext_anskey_env_exec: 79, 2540, 2611, 2611
\__enumext_anskey_env_make:n 63, 77, 1820, 2535,
    2535, 2543
\__enumext_anskey_env_reset_keys: 78, 79, 2580,
    2643
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2535
\__enumext_anskey_env_save_keys: .. 79, 2623,
    2646, 2646
\__enumext_anskey_env_store: .. 80, 2639, 2646,
    2688
\__enumext_anskey_env_unknown:n 78, 2563, 2566
\__enumext_anskey_env_unknown:nn . 2568, 2570
\l__enumext_anskey_level_int .. 28, 2467, 2468
\__enumext_anskey_safe_inner: . 76, 2440, 2446,
    2465
\__enumext_anskey_safe_inner:n ..... 75
\__enumext_anskey_safe_outer: . 75, 2427, 2446,
    2446
\__enumext_anskey_show_wrap_arg:n . 74, 2359,
    2359, 2377, 2392
\__enumext_anskey_show_wrap_left:n 74, 2304,
    2373, 2373
\__enumext_anskey_unknown:n 75, 2395, 2409, 2411
\__enumext_anskey_unknown:nn . 2395, 2413, 2415
\__enumext_anskey_wrapper:n ..... 2000, 2371
\__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, 107, 975,
    975, 3362
\__enumext_before_args_exec_v: 991, 991, 3509
\__enumext_before_args_exec_vii: . 1007, 1007,
    4060
\__enumext_before_args_exec_viii: 1011, 4307
\__enumext_before_env:nn 77, 197, 201, 2488, 2500,
    2512, 2613
\__enumext_before_keys_exec: 46, 975, 979, 3267
\__enumext_before_keys_exec_v: 991, 995, 3455
\__enumext_before_keys_exec_vii ..... 1007
\__enumext_before_keys_exec_viii: . 1015, 4014
\__enumext_before_keys_exec_viiii: 1019, 4263
\__enumext_before_list: ... 94, 3261, 3359, 3359
\__enumext_before_list_v: ... 3450, 3506, 3506
\__enumext_before_list_vii: ... 107, 4009, 4056,
    4056
\__enumext_before_list_viii: .. 112, 4259, 4304,
    4304
\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 89, 3150, 3150,
    3181, 3186, 3228
\__enumext_check_ans_active: . 64, 94, 107, 1856,
    1856, 3363, 4059
\g__enumext_check_ans_item_tl ..... 83
\g__enumext_check_ans_key_bool 65, 66, 144, 332,
    1915, 1921, 2727
\l__enumext_check_ans_key_bool 65, 1841, 1846,
    1912, 1918
\__enumext_check_ans_key_hook: 65, 95, 107, 1909,
    1909, 3440, 4067
\__enumext_check_ans_level: 64, 1856, 1862, 1866
\__enumext_check_ans_log: 65, 66, 81, 1955, 1955,
    2731
\__enumext_check_ans_log_msg_greater: 1955,
    1961, 1974
\__enumext_check_ans_log_msg_less: 1955, 1959,
    1964
\__enumext_check_ans_log_msg_same_ok: 1955,
    1960, 1969
\__enumext_check_ans_msg_greater: 1931, 1937,
    1950
\__enumext_check_ans_msg_less: 1931, 1935, 1940
\__enumext_check_ans_msg_same_ok: 1931, 1936,
    1945
\__enumext_check_ans_show: .. 65, 80, 1931, 1931,
    2729
\l__enumext_check_answers_bool . 63, 64, 75, 85,
    144, 1818, 1845, 1860, 2147, 2171, 2178, 2202, 2429,
    2626, 2850, 2939, 2973, 4172
\__enumext_check_starred_cmd:n 32, 66, 83, 1979,
    1979, 3461, 3638, 4273
\g__enumext_check_starred_cmd_int 144, 1982,
    1988, 1993, 3118, 3608, 4386
\l__enumext_check_start_line_env_tl . 32, 144,
    295, 303, 311, 1985, 1991, 1994
\l__enumext_columns_sep_v_dim 3527, 3529, 3537
\l__enumext_columns_sep_vii_dim .. 3738, 3740,
    3749, 3761, 3837, 4238
\l__enumext_columns_sep_viii_dim . 3769, 3771,
    3780, 3792, 3886, 4481
\l__enumext_columns_v_int 1251, 1399, 3525, 3533,
    3545, 3550
\l__enumext_columns_vii_int .. 3743, 3746, 3750,
    3759, 3801, 3805, 3808, 3814, 3820, 3824, 4233, 4244
\l__enumext_columns_viii_int . 3774, 3777, 3781,
    3790, 3850, 3854, 3857, 3863, 3869, 3873, 4476, 4489
\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_vi_tl ..... 45, 452
\l__enumext_counter_vii_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` . . . [121](#), [4701](#), [4729](#),  
[4735](#), [4749](#)  
`\__enumext_default_item:n` . . . [2935](#), [2935](#), [2999](#)  
`\__enumext_define_counters:Nn` [26](#), [438](#), [438](#), [447](#),  
[448](#), [449](#), [450](#), [451](#), [452](#), [453](#), [454](#)  
`\__enumext_endminipage:` . [33](#), [363](#), [366](#), [380](#), [3699](#),  
[4218](#), [4460](#)  
`\g__enumext_envir_name_tl` [31](#), [34](#), [266](#), [281](#), [340](#),  
[1788](#), [1793](#), [1803](#), [1943](#), [1948](#), [1953](#), [1967](#), [1972](#), [1977](#)  
`\l__enumext_envir_name_tl` . [31](#), [32](#), [34](#), [235](#), [245](#),  
[294](#), [302](#), [310](#), [5097](#), [5100](#), [5107](#), [5110](#), [5117](#), [5120](#),  
[5127](#), [5130](#), [5136](#), [5140](#), [5146](#), [5150](#), [5207](#), [5211](#)  
`\__enumext_execute_after_env:` [32](#), [33](#), [62](#), [65](#), [66](#),  
[76](#), [80](#), [2717](#), [2717](#), [3445](#), [4254](#)  
`\__enumext_fake_item:` . . . . . [899](#), [899](#), [3213](#)  
`\l__enumext_fake_item_indent_v_dim` [918](#), [923](#)  
`\l__enumext_fake_item_indent_v_tl` [920](#), [3101](#),  
[3105](#), [3113](#)  
`\l__enumext_fake_item_indent_vii_dim` [930](#), [935](#)  
`\l__enumext_fake_item_indent_vii_tl` [932](#), [4214](#)  
`\l__enumext_fake_item_indent_viii_dim` . [942](#),  
[947](#), [4452](#)  
`\l__enumext_fake_item_indent_viii_tl` . . [944](#),  
[4450](#), [4455](#)  
`\l__enumext_fake_item_indent_X_tl` . . . . . [98](#)  
`\__enumext_fake_item_vii:` . . . . [899](#), [927](#), [3245](#)  
`\__enumext_fake_item_viii:` . . . . [899](#), [939](#), [3250](#)  
`\__enumext_filter_first_level:n` . . [118](#), [4598](#),  
[4598](#), [4632](#), [4643](#)  
`\__enumext_filter_first_level_key:n` [119](#), [4598](#),  
[4603](#), [4607](#)  
`\__enumext_filter_first_level_pair:nn` . [119](#),  
[4598](#), [4604](#), [4616](#)  
`\__enumext_filter_save_key:n` . . [69](#), [2063](#), [2071](#),  
[2094](#), [2100](#), [2102](#), [2102](#), [4530](#), [4534](#), [4538](#), [4542](#), [4546](#),  
[4550](#)  
`\__enumext_filter_save_key_key:n` . . [69](#), [2102](#),  
[2107](#), [2111](#)  
`\__enumext_filter_save_key_pair:nn` [69](#), [2102](#),  
[2108](#), [2119](#)  
`\__enumext_filter_series:n` [57](#), [1545](#), [1545](#), [1583](#),  
[1595](#), [1600](#)  
`\__enumext_filter_series_key:n` [57](#), [1545](#), [1550](#),  
[1554](#)  
`\__enumext_filter_series_pair:nn` . . [58](#), [1545](#),  
[1551](#), [1562](#)  
`\g__enumext_footnote_arg_seq` . [162](#), [3709](#), [3722](#),  
[3732](#)  
`\g__enumext_footnote_int` . [162](#), [3716](#), [3719](#), [3721](#),  
[3723](#)  
`\g__enumext_footnote_int_seq` . [162](#), [3710](#), [3723](#),  
[3728](#), [3731](#)  
`\__enumext_footnotes_key_bool` . . . . . [34](#)  
`\l__enumext_footnotes_key_bool` [29](#), [34](#), [110](#), [152](#),  
[395](#), [400](#), [409](#), [4181](#), [4229](#), [4424](#), [4471](#)  
`\__enumext_footnotetext:nn` . . . [3703](#), [3703](#), [3733](#)  
`\__enumext_foreach_add_body:n` . [122](#), [4750](#), [4810](#),  
[4813](#)  
`\l__enumext_foreach_after_tl` . . . . . [4754](#), [4822](#)  
`\l__enumext_foreach_before_tl` . . . . [4752](#), [4817](#)  
`\g__enumext_foreach_default_keys_tl` [121](#), [124](#),  
[4772](#), [4793](#)  
`\__enumext_foreach_keyans:nn` . . [122](#), [4750](#), [4789](#),  
[4791](#)  
`\l__enumext_foreach_name_prop_tl` . [124](#), [4795](#),  
[4820](#)  
`\l__enumext_foreach_print_seq` [124](#), [4805](#), [4811](#),  
[4815](#)  
`\l__enumext_foreach_sep_tl` . . . . . [4764](#), [4811](#)  
`\l__enumext_foreach_start_int` . . . . [4756](#), [4807](#)  
`\l__enumext_foreach_step_int` . . . . . [4760](#), [4808](#)  
`\l__enumext_foreach_stop_int` . [4758](#), [4800](#), [4802](#),  
[4809](#)  
`\__enumext_foreach_wrapper:n` . . . . . [4762](#), [4818](#)  
`\__enumext_getkeyans:nn` . . [117](#), [4513](#), [4517](#), [4517](#)  
`\__enumext_getkeyans_aux:n` [116](#), [4501](#), [4504](#), [4504](#)  
`\l__enumext_hyperref_bool` . [29](#), [34](#), [35](#), [152](#), [391](#),  
[412](#), [429](#), [2349](#), [2838](#), [4167](#), [4415](#)  
`\__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](#), [92](#), [107](#), [368](#),  
[368](#), [3296](#), [4030](#)  
`\__enumext_is_not_nested:` [26](#), [31](#), [92](#), [107](#), [229](#), [229](#),  
[3297](#), [4031](#)  
`\__enumext_is_on_first_level:` . [26](#), [31](#), [93](#), [107](#),  
[229](#), [255](#), [3303](#), [4043](#)  
`\g__enumext_item_anskey_int` [75](#), [83](#), [144](#), [327](#), [354](#),  
[355](#), [1928](#), [2298](#), [2852](#)  
`\__enumext_item_answer_diff:` . . [65](#), [66](#), [80](#), [1924](#),  
[1924](#), [2724](#)  
`\g__enumext_item_answer_diff_int` . [65](#), [66](#), [144](#),  
[328](#), [1926](#), [1933](#), [1957](#)  
`\l__enumext_item_column_pos_vii_int` [108](#), [3808](#),  
[3814](#), [3820](#), [3824](#), [3831](#), [4095](#), [4233](#), [4236](#)  
`\l__enumext_item_column_pos_viii_int` . . [113](#),  
[3857](#), [3863](#), [3869](#), [3873](#), [3880](#), [4319](#), [4476](#), [4479](#)  
`\l__enumext_item_column_pos_X_int` . . . . . [165](#)  
`\g__enumext_item_count_all_vii_int` [108](#), [3832](#),  
[4096](#), [4244](#), [4251](#)  
`\g__enumext_item_count_all_viii_int` [113](#), [3881](#),  
[4320](#), [4488](#), [4496](#)  
`\g__enumext_item_count_all_X_int` . . . . . [165](#)  
`\g__enumext_item_number_bool` . . . . . [144](#)  
`\l__enumext_item_number_bool` [64](#), [150](#), [1878](#), [1883](#),  
[1887](#), [1891](#), [1904](#), [2472](#), [2526](#), [2942](#), [2976](#), [4175](#)  
`\g__enumext_item_number_int` [64](#), [65](#), [144](#), [326](#), [353](#),  
[355](#), [1877](#), [1882](#), [1886](#), [1890](#), [1903](#), [1928](#), [2941](#), [2975](#),  
[4174](#)  
`\__enumext_item_peek_args_vii:` [108](#), [4097](#), [4099](#),  
[4099](#)  
`\__enumext_item_peek_args_viii:` . . [113](#), [4321](#),  
[4323](#), [4323](#)  
`\__enumext_item_star_exec:` . [86](#), [2954](#), [2981](#), [3010](#)  
`\l__enumext_item_starred_vii_bool` [4114](#), [4130](#),  
[4185](#)  
`\l__enumext_item_starred_viii_bool` [4338](#), [4354](#),  
[4428](#), [4448](#)  
`\l__enumext_item_starred_X_bool` . . . . . [165](#)  
`\__enumext_item_std:w` . . [33](#), [85](#), [88](#), [100](#), [357](#), [361](#),  
[2945](#), [2951](#), [2979](#), [3101](#), [3105](#), [3113](#), [3675](#)  
`\g__enumext_item_symbol_aux_tl` . [85](#), [128](#), [2959](#),  
[2962](#), [2987](#), [3015](#)  
`\g__enumext_item_symbol_aux_vii_tl` [4138](#), [4187](#),  
[4190](#), [4194](#), [4196](#)



```

\g__enumext_item_symbol_aux_X_tl . . . . . 165
\l__enumext_item_symbol_sep_vii_dim . . 4147,
    4155, 4193, 4195
\l__enumext_item_symbol_vii_tl . . . . . 4190
\l__enumext_item_text_vii_box 4180, 4221, 4228
\l__enumext_item_text_viii_box 4423, 4463, 4470
\l__enumext_item_text_X_box . . . . . 165
\l__enumext_item_width_vii_dim . . . 3747, 3756,
    3835, 3843, 3844
\l__enumext_item_width_viii_dim . . 3778, 3787,
    3884, 3892, 3893
\l__enumext_item_width_X_dim . . . . . 165
\l__enumext_itemindent_X_dim . . . . . 71
\l__enumext_itemsep_i_skip . . . 1181, 1186, 1242,
    1247
\l__enumext_itemsep_ii_skip . . . . . 1190, 1195
\l__enumext_itemsep_iii_skip . . . . . 1199, 1204
\l__enumext_itemsep_vii_skip . . . . . 4250
\l__enumext_itemsep_viii_skip . . . . . 4495
\l__enumext_joined_item_aux_vii_int . . 3829,
    3830, 3831, 3832, 3838
\l__enumext_joined_item_aux_viii_int . . 3878,
    3879, 3880, 3881, 3887
\l__enumext_joined_item_aux_X_int . . . . 165
\__enumext_joined_item_vii:w . . 108, 4102, 4103,
    4105, 4105
\l__enumext_joined_item_vii_int . . 3800, 3801,
    3804, 3806, 3812, 3817, 3822, 3827, 3829, 3835
\__enumext_joined_item_viii:w . . 113, 4326, 4327,
    4329, 4329
\l__enumext_joined_item_viii_int . . 3849, 3850,
    3853, 3855, 3861, 3866, 3871, 3876, 3878, 3884
\l__enumext_joined_item_X_int . . . . . 165
\l__enumext_joined_width_vii_dim . . 3833, 3840,
    3843, 4211, 4223
\l__enumext_joined_width_viii_dim 3882, 3889,
    3892, 4445, 4465
\l__enumext_joined_width_X_dim . . . . . 165
\__enumext_keyans_addto_prop:n 81, 2737, 2737,
    3115, 3605
\__enumext_keyans_addto_seq:n . . 82, 2811, 2811,
    3117, 3607
\__enumext_keyans_addto_seq_link: 2811, 2832,
    2834, 4385
\__enumext_keyans_anspic_code:nnn . . 99, 3596,
    3599, 3599
\__enumext_keyans_default_item:n . . 88, 3096,
    3096, 3132
\l__enumext_keyans_env_bool 34, 3328, 3341, 3489,
    3576
\__enumext_keyans_fake_item: . . 899, 915, 3203
\l__enumext_keyans_level_h_int . . 28, 642, 669,
    2456, 2518, 2789, 4037, 4279, 4280
\l__enumext_keyans_level_int . . 28, 1357, 2452,
    2514, 2784, 3488, 3493, 3590
\__enumext_keyans_make_label: 37, 89, 3120, 3136,
    3201
\__enumext_keyans_mini_right_cmd:n 54, 1359,
    1394, 1394
\__enumext_keyans_mini_set_vskip: . . . . . 51
\__enumext_keyans_minipage_add_space: . . 52,
    1257, 1257, 3518
\__enumext_keyans_minipage_set_skip: . 1222,
    1222, 1259
\__enumext_keyans_multi_addvspace: 1123, 1134,
    3542
\__enumext_keyans_multi_set_vskip: 49, 1123,
    1123, 1136
\__enumext_keyans_multicols_start: 3506, 3521,
    3523
\__enumext_keyans_multicols_stop: 1398, 3506,
    3548, 3574
\__enumext_keyans_name_and_start: 26, 32, 288,
    288, 3490, 3650, 4284
\__enumext_keyans_parse_keys:n 3449, 3502, 3502
\l__enumext_keyans_pic_above_int . . 139, 3690,
    3691, 3693
\l__enumext_keyans_pic_above_skip . . 100, 139,
    3629, 3669
\__enumext_keyans_pic_arg_two: 100, 3627, 3657,
    3657
\l__enumext_keyans_pic_below_int . . 139, 3690,
    3691, 3694
\l__enumext_keyans_pic_body_seq . . 99, 101, 139,
    3594, 3634, 3698
\__enumext_keyans_pic_do:n 101, 3634, 3636, 3682,
    3682, 3686
\l__enumext_keyans_pic_level_int . . 28, 1341,
    2460, 2522, 2740, 2779, 2814, 2902, 3645, 3646
\__enumext_keyans_pic_row:n . . . 101, 3684, 3687,
    3687
\__enumext_keyans_pic_safe_exec: . . 100, 3623,
    3643, 3643
\__enumext_keyans_pic_skip_abs:N . . 100, 3652,
    3652, 3668
\l__enumext_keyans_pic_width_dim . . 139, 3689,
    3696
\__enumext_keyans_redefine_item: . . 89, 3120,
    3120, 3200
\__enumext_keyans_ref: . . . . . 41, 694, 712, 3202
\__enumext_keyans_ref:n . . . . . 40, 691, 694, 694
\__enumext_keyans_safe_exec: . . 3448, 3482, 3482
\__enumext_keyans_set_item_width: . . 96, 3457,
    3465, 3465
\__enumext_keyans_show_ans: . . 2855, 2863, 2882
\__enumext_keyans_show_item_opt: . . 2855, 2870,
    3113, 3619, 4451
\__enumext_keyans_show_left:n . . 88, 2855, 2855,
    3111, 3614
\__enumext_keyans_show_pos: . . 2855, 2867, 2895
\__enumext_keyans_starred_item:n . . 88, 3108,
    3108, 3128
\__enumext_keyans_store_ref: . . 81, 2758, 2758,
    3116, 3606, 4383
\__enumext_keyans_store_ref_aux_i: 82, 2758,
    2770, 2773
\__enumext_keyans_store_ref_aux_ii: 82, 2758,
    2799, 2801
\__enumext_keyans_unknown_keys:n . 3034, 3038,
    3042
\__enumext_keyans_unknown_keys:nn 3034, 3044,
    3046
\__enumext_keyans_wrapper_opt:n . . 2006, 2878
\l__enumext_label_copy_i_tl . . 2264, 2777, 2782,
    2787, 2792
\l__enumext_label_copy_v_tl . . . . . 2787
\l__enumext_label_copy_vi_tl . . . . . 2782
\l__enumext_label_copy_vii_tl 2240, 2251, 2280,
    2777

```



```

\l__enumext_label_copy_viii_tl . . . . . 2792
\l__enumext_label_copy_X_tl . . . . . 154
\l__enumext_label_fill_left_v_tl . . . . . 3140
\l__enumext_label_fill_left_X_tl . . . . . 98
\l__enumext_label_fill_right_v_tl . . . . . 3147
\l__enumext_label_fill_right_X_tl . . . . . 98
\l__enumext_label_font_style_v_tl 3141, 3618
\l__enumext_label_font_style_vii_tl . . . 4199
\l__enumext_label_font_style_viii_tl . . 4433
\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 . . 81, 82, 677, 2745, 2819,
2889, 2929, 3110, 3114, 3452, 3613, 3615
\l__enumext_label_vi_tl . 81, 82, 677, 2742, 2816,
3613, 3615, 3619
\l__enumext_label_vii_tl . 604, 4125, 4150, 4157
\l__enumext_label_viii_tl 604, 4349, 4377, 4381
\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 . . . 2887, 2892, 2900,
2932, 4389, 4404
\l__enumext_labelsep_v_dim . . . . . 3532
\l__enumext_labelsep_vii_dim . 2364, 2887, 2900,
3742, 3752, 3836, 4148, 4209, 4225
\l__enumext_labelsep_viii_dim 3773, 3783, 3885,
4443, 4452, 4467
\l__enumext_labelwidth_i_dim . 2886, 2892, 2899,
2932, 4389, 4404
\l__enumext_labelwidth_v_dim . . . . . 3532
\l__enumext_labelwidth_vii_dim . . . 2364, 2886,
2899, 3742, 3751, 3836, 4202, 4206, 4224
\l__enumext_labelwidth_viii_dim . . 3773, 3782,
3885, 4436, 4440, 4466
\l__enumext_leftmargin_tmp_v_bool . 100, 3659
\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, 1072, 1074, 1076,
1078, 1111, 1113, 1115, 1117, 1121, 1155, 1161, 1170,
1218, 1366, 1370, 1377, 1442, 1444, 1446, 1449, 1456,
1458, 1460, 1463, 2058, 2060, 2062, 2090, 2091, 2093,
2149, 2157, 2161, 2165, 2368, 2369, 2944, 2945, 2949,
2950, 2951, 2959, 2967, 2968, 2971, 2978, 2979, 2983,
2986, 2988, 3006, 3007, 3008, 3011, 3014, 3264, 3266,
3285, 3290, 3334, 3347, 3354, 3365, 3367, 3370, 3371,
3373, 3377, 3384, 3387, 3389, 3391, 3392, 3393, 3394,
3397, 3402, 3408, 3414, 3417, 3419, 3425
\l__enumext_level_h_int 107, 28, 238, 261, 275, 625,
662, 1348, 1874, 1894, 2259, 2492, 2504, 3342, 4032,
4033
\l__enumext_level_int . 92, 28, 207, 248, 260, 276,
370, 1084, 1178, 1347, 1868, 1900, 2236, 2246, 2252,
2258, 2265, 2274, 2279, 2491, 2503, 2719, 3216, 3298,
3299, 3310, 3318, 3332, 3345, 3398, 3497, 3586, 4076,
4086, 4292, 5137, 5141, 5147, 5151
\__enumext_list_arg_two_i: . . . . . 3182
\__enumext_list_arg_two_ii: . . . . . 3182
\__enumext_list_arg_two_iii: . . . . . 3182
\__enumext_list_arg_two_iv: . . . . . 3182
\__enumext_list_arg_two_v: . 89, 3182, 3454, 3660
\__enumext_list_arg_two_vii: . . . . . 3222, 4013
\__enumext_list_arg_two_viii: . . . . . 3222, 4262
\l__enumext_listoffset_v_dim . 3473, 3478, 3534
\l__enumext_listparindent_vii_dim . . . . 4212
\l__enumext_listparindent_viii_dim . . . 4446
\__enumext_log_answer_vars: . 33, 342, 350, 2726
\__enumext_log_global_vars: . 33, 342, 342, 2725
\__enumext_make_label . . . . . 2991
\__enumext_make_label: . . . . . 37, 86, 3002, 3211
\l__enumext_mark_answer_sym_tl 71, 2012, 2215,
2381, 2904, 2917, 4393
\l__enumext_mark_position_str 128, 2016, 2017,
2043, 2044, 2213
\l__enumext_mark_ref_sym_tl . . 2029, 2354, 2846
\l__enumext_meta_path_tl . 124, 4725, 4726, 4728,
4729
\c__enumext_meta_paths_prop . . . . . 120, 4701
\__enumext_mini_addvspace_vii: 53, 1327, 1327,
3910
\__enumext_mini_addvspace_viii: 53, 1327, 1333,
3965
__enumext_mini_env* . . . . . 368
\__enumext_mini_right_cmd:n 54, 1361, 1363, 1363
\__enumext_mini_set_vskip_vii: 52, 1270, 1270,
1329
\__enumext_mini_set_vskip_viii: 52, 1270, 1292,
1335
\__enumext_minipage:w 33, 363, 365, 374, 3696, 4211,
4445
\l__enumext_minipage_active_v_bool 3516, 3539,
3553, 3561
\g__enumext_minipage_active_vii_bool . . 105,
3921, 3926, 3945
\l__enumext_minipage_active_vii_bool . 3906,
3917
\g__enumext_minipage_active_viii_bool 3976,
3981, 4000
\l__enumext_minipage_active_viii_bool 3961,
3972
\g__enumext_minipage_active_X_bool . . . 165
\l__enumext_minipage_active_X_bool . . . . 86
\__enumext_minipage_add_space: . . 51, 94, 1151,
1209, 3375
\g__enumext_minipage_after_skip 86, 1274, 1286,
3943, 3998
\l__enumext_minipage_after_skip . . 50, 95, 86,
1164, 1165, 1183, 1186, 1192, 1195, 1201, 1204, 1224,
1238, 1240, 1244, 1247, 1294, 1307, 1321, 1372, 1401,
3571
\g__enumext_minipage_center_vii_bool . 3930,
3946
\g__enumext_minipage_center_viii_bool 3985,
4001
\g__enumext_minipage_center_X_bool . . . 165
\l__enumext_minipage_hsep_v_dim . . . . . 3514
\l__enumext_minipage_hsep_vii_dim . . . . 3904
\l__enumext_minipage_hsep_viii_dim . . . 3959
\l__enumext_minipage_left_skip 86, 1225, 1272,
1277, 1281, 1295, 1299, 1313, 1331, 1337
\l__enumext_minipage_left_v_dim . . 3512, 3519
\l__enumext_minipage_left_vii_dim 3900, 3912
\l__enumext_minipage_left_viii_dim 3955, 3967

```

```

\l__enumext_minipage_left_X_dim . . . . . 86
\g__enumext_minipage_right_skip 86, 1273, 1278,
    1282, 3929, 3984
\l__enumext_minipage_right_skip . 50, 86, 1153,
    1159, 1164, 1173, 1226, 1227, 1233, 1238, 1254, 1296,
    1303, 1317, 1380, 1408
\l__enumext_minipage_right_v_dim . 1396, 1405,
    3510, 3514
\g__enumext_minipage_right_vii_dim 104, 3908,
    3928, 3948
\l__enumext_minipage_right_vii_dim 104, 3898,
    3903, 3909
\g__enumext_minipage_right_viii_dim . 3963,
    3983, 4003
\l__enumext_minipage_right_viii_dim . 3953,
    3958, 3964
\g__enumext_minipage_right_X_dim . . . . . 165
\g__enumext_minipage_right_X_skip . . . . . 165
\__enumext_minipage_set_skip: . 50, 1151, 1151,
    1211
\g__enumext_minipage_stat_int 94, 86, 1385, 1413,
    3374, 3427, 3432, 3517, 3563, 3568
\l__enumext_miniright_code_vii_box 3937, 3941
\g__enumext_miniright_code_vii_tl 105, 3932,
    3939, 3947
\l__enumext_miniright_code_viii_box . 3992,
    3996
\g__enumext_miniright_code_viii_tl 3987, 3994,
    4002
\l__enumext_miniright_code_X_box . . . . . 165
\__enumext_multi_addvspace: . 49, 95, 1106, 1106,
    3405
\__enumext_multi_set_vskip: 48, 1070, 1070, 1108
\l__enumext_multicols_above_ii_skip . . 1089
\l__enumext_multicols_above_iii_skip . . 1095
\l__enumext_multicols_above_iv_skip . . 1101
\l__enumext_multicols_above_v_skip 1125, 1139,
    1149
\l__enumext_multicols_above_X_skip . . . . . 79
\l__enumext_multicols_below_v_skip 1129, 1143,
    3555
\l__enumext_multicols_below_X_skip . . . . . 79
\g__enumext_multicols_right_X_skip . . . . . 79
\__enumext_multicols_start: 94, 3379, 3381, 3381
\__enumext_multicols_stop: 95, 1368, 3411, 3411,
    3437
\__enumext_nested_base_line_fix: . 43, 93, 107,
    819, 829, 3314, 4053
\__enumext_newlabel:nn 29, 35, 72, 422, 422, 2290,
    2805
\l__enumext_newlabel_arg_one_tl 29, 35, 72, 82,
    154, 2283, 2291, 2353, 2794, 2806, 2844
\l__enumext_newlabel_arg_two_tl 29, 35, 71, 154,
    2239, 2249, 2262, 2277, 2292, 2781, 2786, 2791, 2807
\__enumext_parse_foreach_keys:n . . 4750, 4766,
    4783
\__enumext_parse_foreach_keys:nn . 4750, 4773,
    4785
\__enumext_parse_keys:n 43, 58, 3260, 3305, 3305
\__enumext_parse_keys_vii:n . 43, 58, 4008, 4045,
    4045
\__enumext_parse_keys_viii:n . 4258, 4297, 4297
\__enumext_parse_save_key:n 68, 2083, 2088, 2088
\__enumext_parse_save_key_vii:n 68, 2078, 2088,
    2096
\__enumext_parse_series:n 58, 93, 107, 1571, 1571,
    3313, 4051
\__enumext_parse_store_keys:n . . . . . 93
\l__enumext_parsep_i_skip . . . . . 1087, 1089
\l__enumext_parsep_ii_skip . . . . . 1093, 1095
\l__enumext_parsep_iii_skip . . . . . 1099, 1101
\l__enumext_parsep_vii_skip . . . . . 4213
\l__enumext_parsep_viii_skip . . . . . 4447
\l__enumext_partopsep_v_skip . 1141, 1145, 1235,
    1266
\l__enumext_partopsep_viii_skip . . . . . 1305
\__enumext_phantomsection: 35, 386, 415, 419, 435
\__enumext_previus_level_skip: . . 1167, 1176
\__enumext_print_footnote: . . 3703, 3726, 4231,
    4473
\__enumext_print_keyans_box:NN 71, 2207, 2207,
    2220, 2364, 2367, 2891, 2931, 4389, 4404
\l__enumext_print_keyans_i_tl . . . 4535, 4557
\l__enumext_print_keyans_ii_tl . . 4539, 4558
\l__enumext_print_keyans_iii_tl . . 4543, 4559
\l__enumext_print_keyans_iv_tl . . 4547, 4560
\l__enumext_print_keyans_starred_tl 117, 118,
    128, 4531, 4579
\l__enumext_print_keyans_vii_tl 117, 4551, 4561
\l__enumext_print_keyans_X_tl . . . . . 128
\__enumext_printkeyans:nnn 118, 4562, 4565, 4565
\__enumext_redefine_item: . 86, 2991, 2991, 3210
\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: . . 4023,
    4240, 4240
\__enumext_remove_extra_parsep_viii: . 4272,
    4483, 4483
\__enumext_renew_footnote: . . 3703, 3707, 4183,
    4426
\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 . . 78, 80, 2601,
    2696, 2704
\__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,
    2734
\l__enumext_resume_active_bool 58, 60, 61, 1575,
    1695
\__enumext_resume_counter: . 60, 1693, 1699, 1706
\__enumext_resume_counter:n . 58, 60, 1664, 1669,
    1693, 1693, 1763, 1771
\__enumext_resume_counter_save_ans: . . 60, 61,
    1693, 1704, 1736
\__enumext_resume_counter_series: 60, 61, 1693,
    1702, 1719

```

```

\g__enumext_resume_int ... 61, 1616, 1710, 1711
\__enumext_resume_last:n .. 58, 1571, 1577, 1590
\l__enumext_resume_name_tl 61, 1612, 1620, 1623,
    1639, 1647, 1650, 1696, 1697, 1725, 1732
\__enumext_resume_save_counter: .. 59, 95, 107,
    1603, 1603, 3443, 4070
\__enumext_resume_series:n . 60, 1539, 1660, 1660
\__enumext_resume_starred: . 61, 1540, 1757, 1757
\g__enumext_resume_vii_int 61, 1643, 1715, 1716
\l__enumext_rightmargin_vii_dim .. 3754, 3758,
    3763
\l__enumext_rightmargin_viii_dim . 3785, 3789,
    3794
\__enumext_safe_exec: .. 34, 92, 3259, 3294, 3294
\__enumext_safe_exec_vii: . 34, 4007, 4028, 4028
\__enumext_safe_exec_viii: ... 4257, 4277, 4277
\l__enumext_series_name_tl ..... 60
\l__enumext_series_str .. 59, 93, 107, 1537, 1573,
    1581, 1582, 1584, 1586, 1607, 1610, 1614, 1634, 1637,
    1641, 3309, 4049
\__enumext_set_error:nn ..... 4687, 4697, 4699
\__enumext_set_item_width: . 92, 3269, 3277, 3277
\__enumext_set_parse:n ..... 4671, 4687, 4687
\l__enumext_setkey_tmpa_int ... 119, 4664, 4668
\l__enumext_setkey_tmpa_seq .. 119, 4662, 4672,
    4678, 4680, 4682, 4694
\l__enumext_setkey_tmpa_tl ... 119, 4670, 4674
\l__enumext_setkey_tmpb_seq .. 119, 4663, 4666,
    4670, 4671
\l__enumext_setkey_tmpb_tl 119, 4689, 4691, 4692
\l__enumext_show_answer_bool . 2023, 2047, 2375,
    2861, 2875, 3610, 4387
\__enumext_show_length:nnn .. 45, 223, 223, 4908,
    4909, 4910, 4911, 4912, 4913, 4914, 4915, 4916, 4917,
    4923, 4924, 4925, 4926, 4927, 4928, 4929, 4930, 4931,
    4932
\l__enumext_show_position_bool ... 2026, 2050,
    2379, 2865, 2876, 3611, 4391
\g__enumext_standar_bool 31, 92, 34, 237, 240, 259,
    333, 1605, 1670, 1682, 1708, 1721, 1759, 1899, 1913,
    2244, 2257, 2272, 3329
\l__enumext_standar_bool . 92, 95, 34, 2245, 3301,
    3442, 4042
\l__enumext_standar_first_bool 31, 93, 34, 264,
    832, 1592, 1739, 1801, 1808
\__enumext_standar_item_vii:w . 108, 109, 4110,
    4112, 4112
\__enumext_standar_item_viii:w 113, 4334, 4336,
    4336
\__enumext_standar_ref: .... 39, 577, 597, 3212
\__enumext_standar_ref:n .... 38, 569, 577, 577
\g__enumext_standar_series_tl . 61, 1594, 1595,
    1761, 1764
\__enumext_standar_unknown_keys:n 3074, 3078,
    3082
\__enumext_standar_unknown_keys:nn 3074, 3084,
    3086
\g__enumext_starred_bool 31, 107, 34, 247, 250, 274,
    334, 1632, 1675, 1686, 1713, 1728, 1767, 1873, 1919,
    2235, 2775, 3949
\l__enumext_starred_bool .. 107, 34, 1353, 2273,
    2308, 2314, 2362, 2650, 2655, 2884, 2897, 3302, 4041,
    4069, 4285, 4289
\__enumext_starred_columns_set_vii: .. 3736,
    3736, 4016
\__enumext_starred_columns_set_viii: . 3736,
    3767, 4265
\l__enumext_starred_first_bool 31, 107, 34, 279,
    843, 1597, 1748, 1801, 1808
\__enumext_starred_item:nn ... 2954, 2954, 2997
\__enumext_starred_item_exec: 114, 4379, 4379,
    4430
\__enumext_starred_item_vii:w . 108, 109, 4109,
    4128, 4128
\__enumext_starred_item_vii_aux_i:w .. 4128,
    4133, 4136
\__enumext_starred_item_vii_aux_ii:w . 4128,
    4134, 4139, 4141
\__enumext_starred_item_vii_aux_iii:w 4128,
    4144, 4153
\__enumext_starred_item_viii:w 113, 114, 4333,
    4352, 4352
\__enumext_starred_item_viii_aux_i:w .. 114,
    4352, 4357, 4360
\__enumext_starred_item_viii_aux_ii:w . 114,
    4352, 4358, 4372, 4374
\__enumext_starred_joined_item_vii:n 103, 108,
    3798, 3798, 4107
\__enumext_starred_joined_item_viii:n . 103,
    113, 3798, 3847, 4331
\__enumext_starred_ref: .... 40, 622, 660, 3242
\__enumext_starred_ref:n .... 39, 616, 622, 622
\g__enumext_starred_series_tl . 61, 1599, 1600,
    1769, 1772
\__enumext_starred_unknown_keys:n 3056, 3058,
    3060
\__enumext_starred_unknown_keys:nn 3056, 3062,
    3064
\__enumext_start_from:NNn 41, 719, 719, 732, 754,
    760
\l__enumext_start_i_int ..... 1711, 1723, 1742
\__enumext_start_item_tmp_vii: 106, 4019, 4092,
    4092
\__enumext_start_item_tmp_viii: .. 111, 4268,
    4316, 4316
\__enumext_start_item_vii:w 109, 110, 4120, 4125,
    4150, 4157, 4159, 4159
\__enumext_start_item_viii:w .. 113, 4344, 4349,
    4377, 4407, 4407
\g__enumext_start_line_tl 31, 34, 267, 282, 339,
    1943, 1948, 1953, 1967, 1972, 1977
\__enumext_start_list:nn .. 33, 89, 100, 357, 359,
    3263, 3451, 3624, 4011, 4260
\__enumext_start_mini_vii: 107, 3896, 3896, 4061
\__enumext_start_mini_viii: ... 112, 3951, 3951,
    4308
\__enumext_start_save_ans_msg: 62, 1785, 1785,
    1810
\__enumext_start_store_level: . 93, 3262, 3323,
    3323
\__enumext_start_store_level_vii: 108, 4010,
    4072, 4072
\l__enumext_start_vii_int ... 1716, 1730, 1751
\l__enumext_start_X_int ..... 98
\__enumext_stop_item_tmp_vii: .. 106, 108, 110,
    4018, 4022, 4094, 4161
\__enumext_stop_item_tmp_viii: 111, 113, 4267,
    4271, 4318, 4409

```

```

\__enumext_stop_item_vii: 110, 111, 4161, 4216,
4216
\__enumext_stop_item_viii: 116, 4409, 4458, 4458
\__enumext_stop_list: .. 33, 357, 360, 3273, 3462,
3637, 4024, 4274
\__enumext_stop_mini_vii: 105, 107, 3896, 3915,
4065
\__enumext_stop_mini_viii: 112, 3951, 3970, 4312
\__enumext_stop_save_ans_msg: . 62, 1785, 1790,
2723
\__enumext_stop_store_level: .. 93, 3274, 3323,
3352
\__enumext_stop_store_level_vii: . 108, 4025,
4072, 4082
\l__enumext_store_active_bool 28, 63, 110, 1740,
1749, 1817, 2448, 3327, 3340, 3484, 3492, 3582, 3641,
4074, 4084, 4291
\__enumext_store_active_keys:n .. 68, 93, 2056,
2056, 3320
\__enumext_store_active_keys_vii:n . 68, 107,
2056, 2066, 4052
\__enumext_store_addto_prop:n 69, 81, 2131, 2131,
2139, 2299, 2756, 4382
\__enumext_store_addto_seq:n 70, 83, 2140, 2140,
2144, 2151, 2165, 2173, 2182, 2196, 2204, 2357, 2849
\l__enumext_store_anskey_arg_tl .. 28, 73, 110,
2305, 2310, 2312, 2317, 2324, 2327, 2337, 2342, 2345,
2351, 2357
\__enumext_store_anskey_code:n 72, 75, 80, 2296,
2296, 2441, 2694, 2702
\l__enumext_store_anskey_env_tl .. 28, 79, 110,
2624, 2628, 2634, 2696, 2704
\l__enumext_store_anskey_opt_tl 28, 79, 80, 110,
2625, 2652, 2658, 2665, 2671, 2681, 2691, 2700
\__enumext_store_anskey_safe_outer: .... 75
\g__enumext_store_columns_break_bool . 2548,
2649, 2711
\l__enumext_store_columns_break_bool . 2307,
2397
\l__enumext_store_current_label_tl 28, 81-83,
114, 110, 2739, 2742, 2745, 2752, 2754, 2756, 2813,
2816, 2819, 2825, 2830, 2840, 2849, 4362, 4367, 4368,
4381, 4382, 4384
\l__enumext_store_current_label_tmp_tl . 28,
110, 3110, 3114
\l__enumext_store_current_opt_arg_tl 28, 114,
110, 2859, 2872, 2878, 4370
\__enumext_store_internal_ref: .. 71, 72, 2221,
2221, 2302
\g__enumext_store_item_join_int .. 2551, 2656,
2660, 2712
\l__enumext_store_item_join_int .. 2315, 2319,
2400
\g__enumext_store_item_star_bool . 2553, 2663,
2713
\l__enumext_store_item_star_bool . 2322, 2402
\g__enumext_store_item_symbol_sep_dim 2558,
2678, 2683, 2715
\l__enumext_store_item_symbol_sep_dim 2334,
2339, 2407
\g__enumext_store_item_symbol_tl . 2556, 2669,
2673, 2714
\l__enumext_store_item_symbol_tl . 2325, 2329,
2405
\l__enumext_store_keyans_item_opt_sep_
tl .... 2009, 2750, 2752, 2823, 2827, 4365, 4367
\__enumext_store_level_close: . 70, 2145, 2169,
3356
\__enumext_store_level_close_vii: . 70, 2176,
2200, 4088
\__enumext_store_level_open: 70, 93, 2145, 2145,
3335, 3348
\__enumext_store_level_open_vii: .. 70, 2176,
2176, 4078
\g__enumext_store_name_tl 28, 63, 110, 338, 345,
346, 347, 348, 1793, 1819, 1942, 1947, 1952, 1966,
1971, 1976, 2721
\l__enumext_store_name_tl 28, 62, 64, 110, 1626,
1629, 1653, 1656, 1744, 1753, 1788, 1797, 1798, 1819,
1820, 1821, 1823, 1824, 1826, 1828, 1829, 1831, 1833,
1834, 1858, 2133, 2135, 2142, 2285, 2286, 2387, 2630,
2796, 2797, 2910, 2923, 4399
\l__enumext_store_ref_key_bool 72, 2032, 2300,
2348, 2760, 2837
\l__enumext_store_save_key_vii_bool .. 2068,
2098
\l__enumext_store_save_key_vii_tl 2070, 2071,
2099, 2100, 2180, 2188, 2192, 2196
\l__enumext_store_save_key_X_bool .. 68, 128
\l__enumext_store_save_key_X_tl .... 68, 128
\l__enumext_store_upper_level_X_bool .. 128
\__enumext_storing_exec: . 62, 63, 77, 1795, 1811,
1815
\__enumext_storing_set:n .. 62, 1780, 1795, 1795
\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, 1533,
1544, 1776, 1784, 1837, 1855, 1996, 2037, 2038, 2055,
2074, 2087, 2223, 2230, 2231, 2252, 2265, 2268, 2279,
2762, 2769, 3034, 3041, 3074, 3081, 3182, 3221, 3222,
3256
\__enumext_tmp:nn 487, 508, 509, 540, 541, 556, 749,
774, 855, 877, 878, 898, 951, 959, 960, 974, 1039, 1055,
1056, 1069, 1422, 1438, 3018, 3033
\__enumext_tmp:nnn 557, 573, 574, 575, 576, 604, 620,
621
\__enumext_tmp:nnnnnn 775, 800, 803, 806, 808, 810,
813, 816
\__enumext_tmp:w ..... 4510, 4513
\l__enumext_tmpa_vii_int 3746, 3749, 3758, 3789
\l__enumext_tmpa_viii_int ..... 3777, 3780
\l__enumext_tmpa_X_dim ..... 165
\l__enumext_tmpa_X_int ..... 165
\l__enumext_topsep_v_skip 1127, 1131, 1229, 3640,
3672
\l__enumext_topsep_vii_skip .. 1275, 1284, 1288
\l__enumext_topsep_viii_skip . 1297, 1319, 1323
\__enumext_undefine_anskey_env: . 76, 81, 2481,
2481, 2732
\l__enumext_vspace_a_star_v_bool ..... 1471
\l__enumext_vspace_a_star_vii_bool ... 1493
\l__enumext_vspace_a_star_viii_bool ... 1504
\l__enumext_vspace_a_star_X_bool ..... 98
\__enumext_vspace_above: 55, 94, 1439, 1439, 3361
\__enumext_vspace_above_v: . 56, 1467, 1467, 3508
\l__enumext_vspace_above_v_skip .. 1469, 1473,
1475

```

\__enumext_vspace_above_vii:	56, 107, 1489, 1489, 4058
\l__enumext_vspace_above_vii_skip	1491, 1495, 1497
\__enumext_vspace_above_viii:	56, 1489, 1500, 4306
\l__enumext_vspace_above_viii_skip	1502, 1506, 1508
\l__enumext_vspace_b_star_v_bool	1482
\l__enumext_vspace_b_star_vii_bool	1515
\l__enumext_vspace_b_star_viii_bool	1526
\l__enumext_vspace_b_star_X_bool	98
\__enumext_vspace_below:	55, 95, 1453, 1453, 3441
\__enumext_vspace_below_v:	56, 1478, 1478, 3578
\l__enumext_vspace_below_v_skip	1480, 1484, 1486
\__enumext_vspace_below_vii:	56, 107, 1511, 1511, 4068
\l__enumext_vspace_below_vii_skip	1513, 1517, 1519
\__enumext_vspace_below_viii:	56, 1511, 1522, 4314
\l__enumext_vspace_below_viii_skip	1524, 1528, 1530
\__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	3104
\l__enumext_wrap_label_opt_vii_bool	109, 4119
\l__enumext_wrap_label_opt_viii_bool	113, 4343
\l__enumext_wrap_label_opt_X_bool	98
\l__enumext_wrap_label_v_bool	3100, 3104, 3112, 3142
\l__enumext_wrap_label_vii_bool	109, 4118, 4123, 4131, 4200
\l__enumext_wrap_label_viii_bool	113, 4342, 4347, 4355, 4434
\l__enumext_wrap_label_X_bool	98
\__enumext_wrapper_label_v:n	3144, 3619
\__enumext_wrapper_label_vii:n	4203
\__enumext_wrapper_label_viii:n	4437
\l__enumext_write_aux_file_tl.	29, 72, 82, 154, 2288, 2294, 2803, 2809
enumext*	5, 4005
enumXi	447
enumXii	447
enumXiii	447
enumXiv	447
enumXv	447
enumXvi	447
enumXvii	447
enumXviii	447
Environments provide by enumext:	
anskey*	28, 63, 71, 72, 74, 76, 77, 79, 81, 93, 108, 118, 123, 125
enumext*	25, 26, 29-31, 34, 36, 39-45, 47, 52, 53, 56-62, 64, 65, 67-76, 79, 81, 82, 86, 87, 91-93, 101-103, 105, 108, 110, 111, 113, 115, 117, 118, 120, 124, 127, 128
enumext	25, 26, 30, 31, 34, 36-43, 45-55, 57-62, 64, 65, 67-76, 79, 81, 82, 85, 86, 88-90, 92, 93, 96, 97, 100, 102, 104, 107, 108, 117, 118, 120, 124, 125, 127
keyans*	25, 26, 28-32, 36, 39-45, 47, 52, 53, 56, 63, 66, 67,

	69, 77, 81, 87, 91, 101–103, 112, 124, 126, 128
keyanspic	25, 26, 28, 29, 32, 36, 37, 40, 63, 66, 69, 70, 77, 81–83, 98–100, 126
keyans	25, 26, 28, 29, 31, 32, 36, 37, 40–43, 45, 47, 49, 51–56, 63, 66, 67, 69, 70, 77, 81–83, 87–90, 96–100, 104, 113, 124, 126
Environments:	
list	30, 33, 89, 92
lrbox	102, 110, 111, 115, 116
minipage	30, 33, 34, 47, 49, 50, 98–102, 110, 111, 116
multicols	47–50, 54, 94, 95
scontents	77, 79
exp commands:	
\exp_after:wN	4513
\exp_args:Ne	2693, 2701, 3317, 4501
\exp_args:NV	2413, 2568, 3044, 3062, 3084, 4785
\exp_not:N	58, 478, 592, 637, 654, 707, 908, 922, 923, 934, 935, 946, 947, 2353, 2384, 2385, 2842, 2907, 2908, 2920, 2921, 4396, 4397, 4510
\exp_not:n	269, 284, 297, 305, 313, 531, 551, 592, 593, 637, 638, 654, 655, 707, 708, 909, 1560, 1569, 2020, 2117, 2129, 2291, 2319, 2329, 2339, 2353, 2354, 2660, 2673, 2683, 2806, 2844, 2846, 4614, 4624, 4817, 4822
F	
\fbbox	2003
\fbboxrule	2003
\fbboxsep	2003
file commands:	
\file_input_stop:	5221
first	960
font	487
\footnote	101
footnote	101, 3711
\footnotemark	3721
\footnotesize	2385, 2908, 2921, 4397
\footnotetext	3705
\foreachkeyans	16, 121, 4750
G	
\getkeyans	16, 116, 4499
group commands:	
\group_begin:	2383, 2428, 2603, 2690, 2906, 2919, 4179, 4198, 4395, 4422, 4432, 4556
\group_end:	2390, 2444, 2707, 2913, 2926, 4208, 4220, 4402, 4442, 4462, 4563
H	
\hbadness	4227, 4469
hbox commands:	
\hbox_set:Nn	467
\hfill	517, 521, 526, 527, 1375, 1404, 2353, 2842, 3920, 3975
hook commands:	
\hook_gput_code:nnn	9, 195, 199, 203, 384
\hook_gremove_code:nn	79, 2619
\hook_gset_rule:nnnn	385
\hook_if_empty:nTF	2617
\hspace	4238, 4481
\hyperlink	73, 83
\hyperlink	2353, 2842
\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 ..... 3831, 3880

  \int\_case:nn ... 1084, 1178, 1868, 1894, 1933, 1957

  \int\_compare:nNnTF .. 370, 625, 642, 662, 669, 1169, 1251, 1341, 1357, 1369, 1399, 1981, 1987, 2452, 2456, 2460, 2468, 2514, 2518, 2522, 2719, 2740, 2779, 2784, 2789, 2814, 2902, 3299, 3310, 3332, 3345, 3383, 3398, 3413, 3427, 3493, 3497, 3525, 3550, 3563, 3586, 3590, 3646, 3801, 3811, 3827, 3850, 3860, 3876, 4033, 4037, 4076, 4086, 4233, 4242, 4280, 4292, 4475, 4485, 4668, 4800

  \int\_compare\_p:nNn ... 238, 248, 260, 261, 275, 276, 1347, 1348, 1874, 1900, 2236, 2246, 2258, 2259, 2274, 2315, 2491, 2492, 2503, 2504, 2656, 3342

  \int\_decr:N ..... 3830, 3879

  \int\_eval:n .. 355, 762, 2135, 2286, 2385, 2797, 2908, 2921, 3197, 3241, 3819, 3868, 4397

  \int\_from\_alph:n ..... 727, 741

  \int\_from\_roman:n ..... 729, 743

  \int\_gadd:Nn ..... 3832, 3881

  \int\_gdecr:N ..... 1877, 1882, 1886, 1890, 1903

  \int\_gincr:N 1710, 1715, 2298, 2852, 2941, 2975, 3118, 3374, 3517, 3608, 4096, 4174, 4320, 4386

  \int\_gset:Nn ..... 1926, 3719

  \int\_gset\_eq:NN 1609, 1616, 1622, 1628, 1636, 1643, 1649, 1655, 3716

  \int\_gzero:N . 326, 327, 328, 1385, 1413, 1993, 2712, 3432, 3568, 4251, 4496

  \int\_if\_exist:NNTF 1584, 1620, 1626, 1647, 1653, 1831

  \int\_incr:N 2467, 3298, 3488, 3645, 4032, 4095, 4279, 4319

  \int\_mod:nn ..... 4244, 4487

  \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, 1586, 1834

  \int\_set:Nn 723, 727, 729, 1723, 1730, 1742, 1751, 2604, 3690, 3691, 3746, 3777, 3800, 3806, 3822, 3849, 3855, 3871, 4227, 4469, 4664, 4802

  \int\_set\_eq:NN ..... 1711, 1716, 3829, 3878

  \int\_sign:n ..... 1928

  \int\_step\_function:nnN ..... 2252, 2265, 2279

  \int\_step\_function:nnnN ..... 4806

  \int\_step\_inline:nn ..... 4716

  \int\_step\_inline:nnn ..... 3692

  \int\_to\_roman:n ..... 207, 2232, 2269

  \int\_use:N 348, 353, 354, 1170, 1370, 1725, 1732, 1744, 1753, 3197, 3216, 3241, 3318, 3384, 3393, 3408, 3414, 3804, 3805, 3817, 3853, 3854, 3866, 5137, 5141, 5147, 5151

  \int\_zero:N ..... 4236, 4479

\item . 85, 88, 108, 110, 113, 115, 361, 2153, 2159, 2184, 2190, 2312, 2816, 2819, 2993, 3122, 3677, 4017, 4019, 4266, 4268, 4384

\item\* ..... 5, 14, 66, 3120

item-pos\* ..... 3018

item-sym\* ..... 3018

\itemindent ..... 90

\itemindent ..... 89

itemindent ..... 855

\itemsep ..... 99, 100

\itemsep ..... 3661, 3667

\itemwidth . 437, 2003, 3279, 3288, 3467, 3476, 3840, 3844, 3889, 3893

K

keyans ..... 14, 3446

keyans\* ..... 14, 4255

keyanspic ..... 15, 3621

Keys for \anskey provide by enumext:

  break-col ..... 73, 74, 77-79

  item-join ..... 73, 74, 77-79

  item-pos\* ..... 73, 74, 77, 78, 80

  item-star ..... 73, 74, 77, 78, 80

  item-sym\* ..... 73, 74, 77, 78, 80

Keys for anskey\* provide by enumext:

  break-col ..... 73, 74, 77-79

  item-join ..... 73, 74, 77-79

  item-pos\* ..... 73, 74, 77, 78, 80

  item-star ..... 73, 74, 77, 78, 80

  item-sym\* ..... 73, 74, 77, 78, 80

Keys for environments provide by enumext:

  above\* ..... 27, 55, 56, 94, 107

  above ..... 27, 55, 56, 94, 107, 112

  after ..... 45, 46, 95, 107, 112

  align ..... 27, 37, 86, 89, 110, 123

  base-fix ..... 43, 57, 69, 93, 107, 118

  before\* ..... 45, 46, 94, 107, 112

  before ..... 45, 46

  below\* ..... 27, 55, 56, 95, 107

  below ..... 27, 55, 56, 95, 107, 112

  check-ans 29-31, 62-66, 69, 80, 83, 95, 96, 107, 111, 124

  columns-sep ..... 47, 94

  columns ..... 27, 47, 55, 94

  first ..... 45, 46, 110

  font ..... 37, 86, 89, 110

  item-pos\* ..... 85, 86

  item-sym\* ..... 28, 85, 86

  itemindent ..... 27, 43, 44, 85, 89, 110

  itemsep ..... 42, 91

  labelsep ..... 37, 90, 110

  labelwidth ..... 36-41, 90

  label ..... 26, 36, 38, 41, 102

  lisparindent ..... 91

  list-indent ..... 27, 43, 44, 100

  list-offset ..... 43, 44, 92, 96

  listparindent ..... 43, 110

  mark-ans ..... 67, 69, 74

  mark-pos ..... 67, 123

  mark-ref ..... 67, 69, 71, 73

  mini-env .. 27, 34, 47, 54, 55, 69, 94, 104, 105, 107, 112

  mini-right\* ..... 27, 30, 47, 69, 105, 107

  mini-right ..... 27, 30, 47, 53, 69, 105, 107

  mini-sep ..... 27, 47, 69, 94

  no-store ..... 29, 62-64, 69, 75, 85

  noitemsep ..... 42

  nosep ..... 42

  parindent ..... 91

  parsep ..... 42, 91, 110

  partopsep ..... 42

  ref ..... 26, 30, 38-40, 124

  resume\* ..... 26, 57, 58, 61-63, 69, 95, 107, 119

  resume ..... 26, 33, 57-63, 69, 95, 107, 119

  rightmargin ..... 43, 102

  save-ans 28, 33, 58-62, 64, 65, 68-70, 75-77, 80-82, 88, 96, 98, 113, 114, 116, 117, 119, 124

  save-key ..... 28, 58, 68, 93, 107

  save-pos ..... 69

  save-ref ..... 29, 35, 67, 69, 71-73, 81, 83, 88, 114

  save-sep ..... 67, 69, 114

series	26, 57–61, 69, 93, 95, 107, 119
show-ans	67, 69, 71, 72, 74, 88, 114
show-length	31, 45, 124
show-pos	28, 67, 71, 72, 74, 83, 88, 114
start*	27, 41, 58
start	27, 30, 41, 58
store-key	68
topsep	42
widest	26, 30, 41
wrap-ans	35, 67, 69, 71, 74
wrap-label*	27, 37, 85, 86, 89, 109, 110, 113
wrap-label	27, 37, 85, 86, 89, 109, 110, 113
wrap-opt	67, 69
keys commands:	
\keys_define:nn	489, 511, 543, 559, 606, 677, 751, 777, 821, 857, 880, 953, 962, 1041, 1058, 1424, 1535, 1778, 1839, 1998, 2040, 2076, 2081, 2395, 2546, 2582, 3020, 3036, 3056, 3076, 4527, 4626, 4742, 4750
\keys_if_exist_p:nn	4738, 4739
\l_keys_key_str	75, 78, 2413, 2568, 3044, 3062, 3084, 4785, 4893
\keys_precompile:nnN	117, 191, 191, 4529, 4533, 4537, 4541, 4545, 4549, 4768
\keys_set:nn	503, 837, 848, 1064, 1429, 1434, 1672, 1677, 1764, 1772, 2433, 3312, 3317, 3504, 4050, 4301, 4581, 4588, 4630, 4635, 4636, 4637, 4638, 4641, 4646, 4647, 4648, 4649, 4650, 4651, 4652, 4684, 4794
\keys_set_known:nn	2700
keyval commands:	
\keyval_parse:NNn	1549, 2106, 4602

L

label	557, 604, 677
Labels provide by enumext:	
\Alph*	36
\Roman*	36
\alph*	36
\arabic*	30, 36
\roman*	36
\labelsep	100
\labelsep	3662, 3665
labelsep	487
\labelwidth	36, 100
\labelwidth	3662, 3663
labelwidth	487
\leftmargin	90
\leftmargin	89, 3662
legacy commands:	
\legacy_if:nTF	4162, 4165, 4410, 4413
\legacy_if_gset_false:n	375
\legacy_if_set_false:n	4164, 4412
\legacy_if_set_true:n	4124, 4149, 4156, 4169, 4348, 4376, 4417
\linewidth	94
\linewidth	3281, 3369, 3469, 3514, 3689, 3749, 3780, 3902, 3957
\list	359
list-indent	855
list-offset	855
\listparindent	3664
listparindent	855
\lrbox	4180, 4423

M

\makebox	102
----------	-----

\makebox	2211, 2213, 2987, 4194, 4202, 4206, 4436, 4440
\makelabel	85, 86, 89, 101
\makelabel	85, 88, 3004, 3138
\makesavenoteenv	408
mark-ans	1996
mark-pos	1996, 2038
mark-ref	1996
mini-env	1039
mini-sep	1039
\minipage	365
\miniright	10, 53, 1339, 1389, 1417, 3430, 3566
mode commands:	
\mode_if_math:TF	2476, 2530
\mode_if_vertical:TF	1109, 1137, 1157, 1212, 1231, 1260
\mode_leave_vertical:	835, 846, 908, 922, 934, 946, 2209, 2985, 4192
msg commands:	
\msg_error:nn	1391, 1419, 2437, 2470, 2474, 2528, 2636, 3495, 3499, 3588, 3648, 3679, 4035, 4282, 4294, 4653, 4712
\msg_error:nnn	582, 629, 646, 699, 1343, 1350, 1355, 1387, 1415, 1684, 1688, 1803, 2419, 2478, 2496, 2508, 2516, 2520, 2524, 2532, 2574, 3050, 3068, 3090, 4039, 4287, 4515, 4524, 4595, 4700, 4731, 4740, 4777, 4798
\msg_error:nnnn	2422, 2450, 2454, 2458, 2462, 2577, 3053, 3071, 3093, 3486, 3584, 3592, 4576, 4780
\msg_error:nnnnn	530, 550, 2019
\msg_fatal:nn	3300
\msg_fatal:nnn	441
\msg_info:nnn	13, 16, 21, 24, 390, 404
\msg_line_context:	4858, 4863, 4868, 4897, 4902, 4907, 4922, 4937, 4941, 4945, 4949, 4953, 4957, 4964, 4971, 4977, 4991, 4995, 5000, 5004, 5008, 5012, 5017, 5021, 5025, 5029, 5034, 5069, 5073, 5078, 5083, 5087, 5092, 5168, 5172, 5177, 5182, 5187, 5191, 5195, 5199, 5203, 5207, 5211, 5215, 5219
\msg_log:nnn	1823, 1828, 1833
\msg_log:nnnnn	352, 1966, 1971, 1976
\msg_log:nnnnnn	344
\msg_new:nnn	4825, 4829, 4833, 4837, 4842, 4855, 4860, 4865, 4870, 4879, 4887, 4891, 4895, 4900, 4905, 4920, 4935, 4939, 4943, 4947, 4951, 4955, 4959, 4968, 4974, 4980, 4984, 4988, 4993, 4998, 5002, 5006, 5010, 5015, 5019, 5023, 5027, 5032, 5067, 5071, 5076, 5081, 5085, 5090, 5166, 5170, 5175, 5180, 5185, 5189, 5193, 5197, 5201, 5205, 5209, 5213, 5217
\msg_new:nnnn	4846, 5037, 5046, 5055, 5061, 5094, 5104, 5114, 5124, 5134, 5144, 5154, 5160
\msg_term:nnnn	1787, 1792, 3206, 3216, 3247, 3252
\msg_term:nnnnn	1947
\msg_warning:nn	3429, 3565
\msg_warning:nnnn	1984, 1990, 3154, 3159, 3803, 3816, 3852, 3865
\msg_warning:nnnnn	1942, 1952
\multicolsep	94
\multicolsep	1173, 1254, 3404, 3541
N	
\NeedsTeXFormat	3
\newcounter	444
\NewDocumentCommand	1339, 2425, 3580, 4499, 4554, 4660, 4709, 4787
\NewDocumentEnvironment	3257, 3446, 3621, 4005, 4255
\newenvsc	2539



<code>\newlabel</code>	35
<code>\newlabel</code>	426
<code>no-store</code>	1837
<code>\noindent</code>	3911, 3966, 4018, 4235, 4267, 4478
<code>\nointerlineskip</code>	1214, 1217, 1262, 1265, 1379, 1407, 3911, 3966
<code>noitemsep</code>	775
<code>\nopagebreak</code>	1120, 1148, 1214, 1217, 1262, 1265, 1330, 1336
<code>\normalfont</code>	2384, 2907, 2920, 4396
<code>nosep</code>	775

P

Packages:	
<code>caption</code>	105
<code>enumext</code>	25, 35, 38, 62, 90, 98, 123
<code>enumitem</code>	35, 36
<code>expl3</code>	101
<code>footnotehyper</code>	34
<code>hyperref</code>	29, 30, 34, 35, 73, 83, 110, 122
<code>lua-visual-debug</code>	50
<code>multicol</code>	25, 122
<code>scontents</code>	25, 76, 77
<code>shortlst</code>	101
<code>\par</code>	1120, 1148, 1217, 1265, 1330, 1336, 1379, 1407, 2361, 3419, 3555, 3571, 3701, 3929, 3943, 3984, 3998, 4235, 4249, 4478, 4494
<code>\parbox</code>	2003
<code>\parindent</code>	4212, 4446
<code>\parsep</code>	48, 99, 100
<code>\parsep</code>	3238, 3661, 3668, 3673
<code>parsep</code>	775
<code>\parskip</code>	4213, 4447
<code>\partopsep</code>	100
<code>\partopsep</code>	3239, 3666
<code>partopsep</code>	775
peek commands:	
<code>\peek_meaning:NTF</code>	4101, 4115, 4132, 4143, 4325, 4339, 4356
<code>\peek_meaning_remove:NTF</code>	4108, 4332
<code>\peek_remove_spaces:n</code>	3126
<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, 117, 4554
prop commands:	
<code>\prop_const_from_keyval:Nn</code>	4701
<code>\prop_count:N</code>	346, 2135, 2286, 2387, 2797, 2910, 2923, 4399, 4803
<code>\prop_get:NnNTF</code>	4727
<code>\prop_gput_if_not_in:Nnn</code>	2133
<code>\prop_if_exist:NTF</code>	1821, 4519, 4796
<code>\prop_item:Nn</code>	4521, 4820
<code>\prop_new:N</code>	1824
<code>\ProvidesExplPackage</code>	4

R

<code>\raggedcolumns</code>	3407, 3544
<code>\ref</code>	71, 81
<code>ref</code>	557, 604, 677
<code>\refstepcounter</code>	4171, 4419

regex commands:	
<code>\regex_match:nnTF</code>	211, 726, 728, 740, 742, 2632
<code>\regex_replace_once:nnN</code>	219
<code>\renewcommand</code>	592, 637, 654, 707
<code>\RenewDocumentCommand</code>	1389, 1417, 2993, 3004, 3122, 3138, 3677, 3711
<code>\RequirePackage</code>	17, 25
<code>resume</code>	1533
<code>resume*</code>	1533
<code>rightmargin</code>	855
<code>\Roman</code>	36, 41
<code>\Roman</code>	463
<code>\roman</code>	36, 41
<code>\roman</code>	464, 575, 4544

S

<code>\s</code>	2633
<code>save-ans</code>	1776
<code>save-key</code>	2074
<code>save-ref</code>	1996
<code>save-sep</code>	1996
scan commands:	
<code>\scan_stop:</code>	100, 3675, 4017, 4266, 4510, 4513
scontents internal commands:	
<code>\l_scontents_fname_out_tl</code>	2592
<code>\_scontents_parse_environment_keys:n</code>	2598
<code>\_scontents_rescan_tokens:n</code>	2605
<code>\l_scontents_storing_bool</code>	2590
<code>\l_scontents_writing_bool</code>	2591
seq commands:	
<code>\seq_clear:N</code>	4662, 4805
<code>\seq_const_from_clist:Nn</code>	4655
<code>\seq_count:N</code>	347, 3634, 4666
<code>\seq_gclear:N</code>	3709, 3710
<code>\seq_gput_right:Nn</code>	2142, 3722, 3723
<code>\seq_if_empty:NTF</code>	3728, 4569, 4680
<code>\seq_if_exist:NTF</code>	1826, 4567
<code>\seq_if_in:NnNTF</code>	4574
<code>\seq_item:Nn</code>	2630, 3698
<code>\seq_map_function:NN</code>	4671
<code>\seq_map_inline:Nn</code>	4582, 4589, 4681, 4682
<code>\seq_map_pairwise_function:NNN</code>	3730
<code>\seq_new:N</code>	122, 123, 125, 139, 163, 164, 1829
<code>\seq_pop_left:NN</code>	4670
<code>\seq_put_right:Nn</code>	3594, 4678, 4694, 4815
<code>\seq_set_from_clist:Nn</code>	4663
<code>\seq_set_map_e:NNn</code>	4672
<code>\seq_show:N</code>	4571
<code>\seq_use:Nn</code>	191, 192, 4811
<code>series</code>	1533
<code>\setcounter</code>	737, 741, 743, 3197, 3241, 3639
<code>\setenumext</code>	6, 118, 4660
<code>\setenumextmeta</code>	6, 120, 4701
<code>show-ans</code>	1996, 2038
<code>show-length</code>	951
<code>show-pos</code>	2038
skip commands:	
<code>\skip_add:Nn</code>	1089, 1095, 1101, 1111, 1115, 1139, 1143, 1159, 1183, 1192, 1201, 1233, 1244, 3661
<code>\skip_gset:Nn</code>	1278, 1282, 1286
<code>\skip_gzero_new:N</code>	1273, 1274
<code>\skip_horizontal:N</code>	923, 935, 947, 4195, 4209, 4443
<code>\skip_horizontal:n</code>	909, 2210, 2218, 2986, 2988, 4193, 4452

<code>\skip_if_eq:nnTF</code>	1087, 1093, 1099, 1165, 1181, 1190, 1199, 1240, 1242, 1275, 1297, 1441, 1455, 1469, 1480, 1491, 1502, 1513, 1524
<code>\skip_new:N</code>	... 81, 82, 83, 87, 88, 89, 90, 91, 143, 183
<code>\skip_set:Nn</code>	1072, 1076, 1125, 1129, 1153, 1227, 1277, 1281, 1299, 1303, 1307, 1313, 1317, 1321, 3655, 3669
<code>\skip_set_eq:NN</code>	1164, 1173, 1238, 1254, 3195, 3237, 3238, 4212, 4213, 4446, 4447
<code>\skip_sub:Nn</code>	... 1186, 1195, 1204, 1247
<code>\skip_use:N</code>	1074, 1078, 1113, 1117, 1121, 1141, 1145, 1155, 1161, 1442, 1446, 1449, 1456, 1460, 1463, 3419
<code>\skip_vertical:N</code>	... 376, 379, 1372, 1401
<code>\skip_zero:N</code>	1172, 1218, 1224, 1225, 1226, 1253, 1266, 3239, 3404, 3541, 3666, 3667
<code>\skip_zero_new:N</code>	... 1272, 1294, 1295, 1296
<code>\c_zero_skip</code>	376, 379, 1087, 1093, 1099, 1165, 1181, 1190, 1199, 1240, 1242, 1275, 1297, 1442, 1456, 1469, 1480, 1491, 1502, 1513, 1524
<code>\small</code>	... 4532, 4536, 4540, 4544, 4548, 4552
<code>\star</code>	... 3024
<code>start</code>	... 749
<code>start*</code>	... 749
<code>\stepcounter</code>	... 3601, 3715
str commands:	
<code>\c_backslash_str</code>	2478, 4858, 4863, 4868, 4873, 4875, 4877, 4882, 4884, 4982, 4986, 4990, 5000, 5004, 5012, 5013, 5017, 5029, 5030, 5034, 5035, 5056, 5058, 5062, 5064, 5092, 5155, 5157, 5161, 5163, 5172, 5173, 5177, 5182, 5183, 5187, 5191, 5195
<code>\c_colon_str</code>	... 2285, 2796, 4510
<code>\c_left_brace_str</code>	... 4963, 4970, 4976
<code>\c_right_brace_str</code>	... 4963, 4970, 4976
<code>\str_case:nn</code>	... 231, 290
<code>\str_case:nnTF</code>	... 1556, 1564, 2113, 2121, 4609, 4618
<code>\str_clear:N</code>	... 3309, 4049
<code>\str_count:n</code>	... 226
<code>\str_if_empty:N</code>	... 1573, 1614, 1641
<code>\str_if_eq:nnTF</code>	... 3198, 3243, 4711
<code>\str_if_in:nnTF</code>	... 4506
<code>\str_new:N</code>	... 129, 178
<code>\str_set:Nn</code>	... 546, 547, 548, 2016, 2017, 2043, 2044
<code>\string</code>	... 408
<code>\strutbox</code>	... 1183, 1192, 1201, 1220, 1244, 1268, 1277, 1278, 1281, 1288, 1301, 1309, 1315, 1323, 3671

T

TeX and  $\LaTeX$  commands:

<code>\@auxout</code>	... 424
<code>\@currentenv</code>	... 231, 290
<code>\protected@write</code>	... 424

tex commands:

<code>\tex_newlinechar:D</code>	... 2604
---------------------------------	----------

text commands:

<code>\text_expand:n</code>	... 4502
<code>\textasteriskcentered</code>	... 2013, 2030
<code>\thepage</code>	... 430

tl commands:

<code>\c_space_tl</code>	2878, 4907, 4922, 4945, 4949, 5136, 5137, 5146, 5147, 5207, 5211
<code>\tl_clear:N</code>	... 516, 522, 1994, 2060, 2070, 2091, 2099, 2305, 2624, 2625, 2739, 2813, 4362
<code>\tl_clear_new:N</code>	... 473
<code>\tl_const:Nn</code>	... 50, 457
<code>\tl_gclear:N</code>	... 338, 339, 340, 1594, 1599, 2714, 3015, 3947, 4002, 4196

<code>\tl_gclear_new:N</code>	... 1581
<code>\tl_gput_right:Nn</code>	... 458
<code>\tl_greplace_all:Nnn</code>	... 479
<code>\tl_gset:Nn</code>	266, 267, 281, 282, 1582, 1595, 1600, 1819, 2628, 2962, 4138
<code>\tl_gset_eq:NN</code>	... 475, 2958, 4189
<code>\tl_if_blank:nTF</code>	2417, 2435, 2572, 3048, 3066, 3088, 4187, 4775
<code>\tl_if_empty:N</code>	... 580, 599, 627, 644, 664, 671, 697, 714, 1607, 1612, 1634, 1639, 1697, 1761, 1769, 1798, 1858, 2149, 2180, 2325, 2669, 2691, 2721, 2750, 2823, 2872, 2983, 4365, 4692
<code>\tl_if_empty:nTF</code>	... 1662
<code>\tl_if_exist:N</code>	... 1667
<code>\tl_if_novalue:nTF</code>	... 2431, 2747, 2821, 2857, 2937, 2956, 2964, 3098, 3307, 3632, 3713, 4047, 4299, 4363
<code>\tl_map_inline:Nn</code>	... 217, 476
<code>\tl_new:N</code>	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
<code>\tl_put_left::Ne</code>	... 2658
<code>\tl_put_left:Nn</code>	2157, 2188, 2310, 2652, 2665, 2671, 2681, 2889, 2929, 3932, 3987, 4381, 4384
<code>\tl_put_right:Nn</code>	474, 590, 635, 652, 705, 2161, 2192, 2239, 2249, 2262, 2277, 2283, 2288, 2312, 2317, 2324, 2327, 2337, 2342, 2345, 2351, 2742, 2745, 2752, 2754, 2781, 2786, 2791, 2794, 2803, 2816, 2819, 2825, 2830, 2840, 4367, 4368
<code>\tl_remove_all:Nn</code>	... 4691
<code>\tl_remove_once:Nn</code>	... 2227, 2766
<code>\tl_replace_all:Nnn</code>	... 478, 4726
<code>\tl_reverse:N</code>	... 2226, 2228, 2765, 2767
<code>\tl_set:Nn</code>	... 58, 235, 245, 294, 295, 302, 303, 310, 311, 443, 517, 521, 526, 527, 579, 624, 696, 906, 920, 932, 944, 1696, 1797, 2061, 2071, 2092, 2100, 2381, 2592, 2859, 2904, 2917, 4370, 4393, 4689, 4725, 4795
<code>\tl_set_eq:NN</code>	484, 585, 588, 632, 634, 649, 651, 702, 704, 2225, 2764, 2777, 3110, 3114, 3613, 3615
<code>\tl_to_str:n</code>	... 1667, 1673, 1678, 4502
<code>\tl_trim_spaces:n</code>	... 474, 4678, 4689, 4695, 4711
<code>\tl_use:N</code>	480, 483, 601, 666, 673, 716, 977, 981, 985, 989, 993, 997, 1001, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 2215, 2232, 2240, 2251, 2264, 2269, 2280, 2945, 2951, 2979, 3006, 3007, 3014, 3101, 3105, 3113, 3140, 3141, 3147, 3264, 3452, 3618, 3939, 3994, 4199, 4210, 4214, 4433, 4444, 4450, 4455, 4557, 4558, 4559, 4560, 4561, 4579, 4674, 4793

token commands:

<code>\token_to_str:N</code>	... 426
<code>topsep</code>	... 775
<code>\topskip</code>	... 1172, 1253
<code>\typeout</code>	... 394, 397, 407, 408

U

<code>\u</code>	... 220, 2633
<code>unknown</code>	... 3034, 3056, 3074
<code>\unskip</code>	... 3433, 3569
use commands:	
<code>\use:N</code>	... 227, 3011, 3266
<code>\use:n</code>	... 1547, 2104, 4508, 4600
<code>\use_none:nn</code>	... 418, 4732
<code>\usecounter</code>	... 3196, 3240

V

<code>\value</code>	... 1610, 1616, 1623, 1629, 1637, 1643, 1650, 1656
---------------------	--

vbox commands:	
\ vbox_set_top:Nn	3937, 3992
\ v space	836, 847, 1446, 1449, 1460, 1463, 1473, 1475, 1484, 1486, 1495, 1497, 1506, 1508, 1517, 1519, 1528, 1530, 3629, 3640, 4250, 4495
<b>W</b>	
widest	749
<b>Z</b>	
wrap-ans	1996
wrap-label	487
wrap-label*	487
wrap-opt	1996
<b>Z</b>	
\ z	2633